THE BE G COMPUTER PROJECTS MAGAZINE FEBRUARY 1984

« LIGHT PENS TO BE WON »

ELECTRONICS & COMPUTING MONTHLY

6 BOOKLIST 7
12 CATALOGUE
16
21 HACKING

43
BOOKS

ROBOT
BOOKLET

45

49

65

81

AN EMAP PUBLICATION

Germany D5.80
U.S.A. $2.95
85p

Your ROBOT
BRITAINS FIRST ROBOTICS MAGAZINE WINTER 1984
Economatics BBC Buggy on test
ROBOT ARM MICROGRASP
Full construction details inside
A history of robotics Robots of the future
CODE NAME
NOW OPERATIONAL
SPECTRAVIDEO VERSUS ATARI

FREE INSIDE

A new magazine for you and your robot

HACKERS' GUIDE TO MICRONET

EXCLUSIVE

HIDDEN DEPTHS OF SPECTRUM STREAMS AND CHANNELS

UNLEASH THE ELECTRON WITH OUR RS432 INTERFACE

MTX 500 - A RIVAL TO THE BBC?

SPECTRUM MICRO PROLOG - MOVE OVER BASIC

BUILD A BBC SIDEWAYS ROM BOARD

Electronics &
Computing Monthly
Scriptor Court,
155 Farringdon Road,
London,
EC1R 3AD

EDITORIAL
01-833-0846
Editor
Gary Evans

Assistant Editor
William Owen

Administration
Liz Gregory

ADVERTISING
01-833-0531

Manager
Claire Fullerton

Executive
Richard Jansz

Classified
Andrew King

PRODUCTION
01-833-0531

Art Editor
Jeremy Webb

Make-up
Time Graphics

Publisher
Alfred Rolington

Distribution
EMAP
National Publications

Published By
EMAP Business And
Computer Publications

Printed by
Riverside Press,
England

Subscriptions
Electronics &
Computing Monthly,
(Subscription
Department),
Competition House,
Farndon Road,
Market Harborough,
Leicestershire.

ABC
MEMBER OF THE AUDIT
BUREAU OF CIRCULATION

Vol. 4

IN THIS ISSUE

Issue 2

See page 77 for details of this month's competition with three RH lightpens as prizes.

Your free, pull out and keep, issue of Your Robot appears between pages 42 and 43.

Electronics & Computing Monthly is normally published on the 13th day of each month

© copyright EMAP Business & Computer Publications Limited 1984. Reasonable care is taken to avoid errors in this magazine however, no liability is accepted for any mistakes which may occur. No material in this publication may be reproduced in any way without the written consent of the publishers. Subscription rates: UK £10.70 incl. post. For overseas rates apply to Subscription Dept., Competition House, Farndon Road, Market Harborough, Leicestershire. Back issues available from: EMAP National Publications (*E&CM* Back Numbers), Bretton Court, Peterborough, PE3 8DZ. Phone 0733 264666.

EDITORIAL

Christmas past seems to have been a fairly merry occasion for the various high street multiples that jumped on the computer bandwagon for the first time last year. For these companies who are in the business of 'shifting boxes' the number of computers going out the door seems to have exceeded all expectations. Indeed, the general manager of one chain of stores claimed sales of some machines were running at eight to ten times the target level.

All well and good, the retailers are happy and the pleasures of Home Computing are being sampled by a larger section of the public. What happens when things start to go wrong though. The level of reliability of at least two of the more popular computers still leaves a lot to be desired and, bearing in mind the high level of sales over the Christmas period, it is not beyond the realms of probability that a significant number of people will experience problems with their computer as we enter 1984.

Even with the most co-operative of retailers, the inconvenience of having to return machines, explain the malfunction, and either accept a replacement or await for their machine to be serviced, is hardly a good start in computing.

The British TV industry of a few years ago also suffered for problems with reliability and during this period, the Japanese, with their reputation for quality, saw sales of their products increase. While the parallel is obviously not an exact one, it does seem that British companies tend to rely on their technical innovation and leave quality control to take care of itself. It's possible to have both though as far eastern companies have proved with, for example, domestic video recorders. Let's hope we British can take a leaf out of their book.

Big Money Prizes

Home computing and the related field of producing and marketing software is starting to attract big money. Within a few days of this issue of *E&CM* appearing, details of a major international software competition are due to be announced. The value of the prizes on offer looks more like a telephone number and the winners of the competition will appear in a televised final.

The categories include arcade games, adventure programs and educational software. We'll have more details in our next issue, but in the meantime, anybody with programming ideas should sit down with their assembler — it will be a very prosperous new year for some.

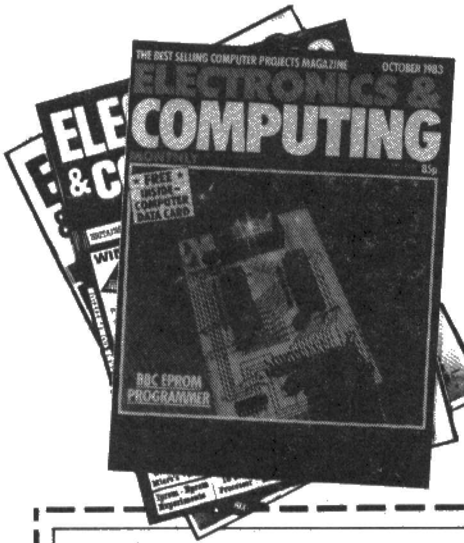
Doing it in style

This month we've changed the way in which we present the projects and features that go to make up *E&CM.* We're pleased with the result but would like to hear from anybody with any comments to make on the new style.

GARY EVANS

Your Robot

***E&CM* is playing host to a brand new Robotics magazine this month. Your Robot as we announced in our January issue is presented free with this issue of *E&CM.* Because of the interest shown in Your Robot the magazine will however be on sale separately in the near future.**



DON'T SUBSCRIBE TO ELECTRONICS AND COMPUTING IF YOU WANT TO . . .

spend hours trudging from newsagent to newsagent only to be told they've sold out of your favourite computer magazine – if you're happy to risk missing a vital part of one of our popular series of articles – if you want to miss the first rate projects that we present every month – if you don't want to be protected from any future increases in cover price.

ON THE OTHER HAND . . .

You could play safe and subscribe by filling in the coupon below and making sure of your copy each and every month.

SEND TO

ELECTRONICS & COMPUTING MONTHLY
SUBSCRIPTIONS DEPT
COMPETITION HOUSE
FARNDON ROAD
MARKET HARBOROUGH
LEICESTERSHIRE
Enquiries: Phone 0733 264666

Please send Electronics & Computing Monthly for the next 12 issues to commence from . . . . .Issue.

I have enclosed a cheque/Postal order for £10.70 (UK only); £15.00 (Overseas, surface); £26.00 (Europe, Air Mail); other rates on request.

Name . . . .

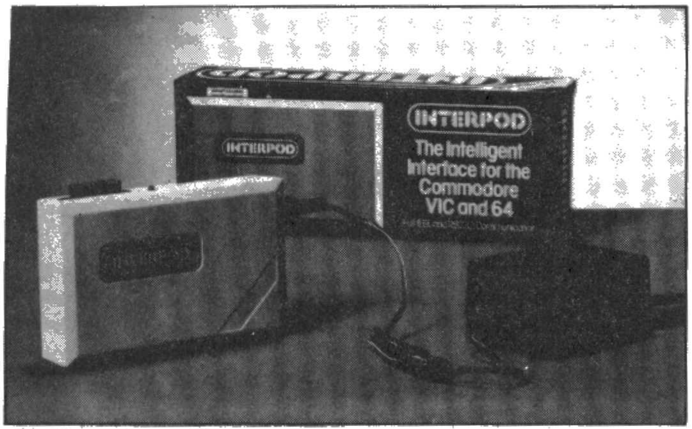
Address . . . .

Town . . . .

Payment accepted by Cheque, Postal Order, International Money Order, Sterling Draft, Access or Visa.

NEW PRODUCTS

NEW PRODUCTS



CBM64 Interface

Not a new product, but worthy of mention is the price reduction by Oxford Computer Systems of Interpod – their popular interface system for the Commodore 64 – to £99.95. Interpod is an intelligent interface that provides the Commodore 64 with both RS232 and IEEE interface capabilities to take advantage of the wide range of peripherals such as dual disk drives and daisy-wheel printers.

One of the major advantages of Interpod is that unlike other systems, it does not use the cartridge port of the Commodore 64. Thus users are still able to run any of their existing cartridge-based programs. *Oxford Computer Systems, Hensington Road, Woodstock, Oxford.*

Joystick Interface

Downsway Electronics have introduced a fully-programmable memory joystick interface for use with the Spectrum.

The interface is fully cased in sturdy injection-moulded ABS plastic styled to match the finish of the computer. The unit plugs directly into the rear expansion port of the Spectrum, with no need for any other connections or software, and a simple two-position switch selects either "program" or "play" modes.

Programming is fast and simple – it is only necessary to press the designated key and, keeping it depressed, to move the joystick to the appropriate position. Any key specified by the game software can be used, and the command is retained in the memory until the computer is switched off or the interface is re-programmed.

Eight directions, (up, down, left, right and diagonals), plus fire can be programmed, and an added feature is that in use the response is immediate.

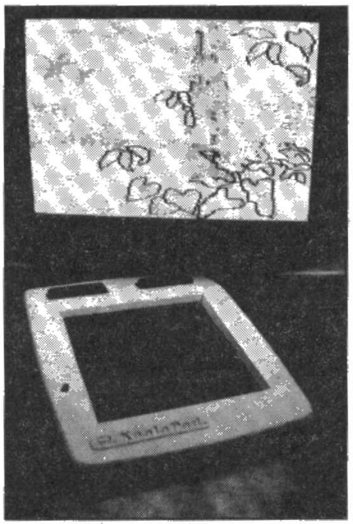
The Downsway interface is compatible with suitable switch-type joysticks such as the Atari, Competition-Pro, Quickshot, Starfighter, Wico etc; it is supplied with complete instructions and is priced at £22.95 including VAT. *Downsway Electronics (UK) Limited, Depot Road, Epsom, Surrey KT17 4RJ.*

CBM graphics tablet

Software specialists Audiogenic have turned their attention to the peripherals market with the launch of Koala Painter – a graphics tablet for the Commodore 64. The Audiogenic package consists of a graphics tablet with separate stylus, disk-based software, and an instruction manual.

The Koalapad, which interfaces directly into the 64, is relatively small (9" x 6" x 1") and lightweight (1lb) so that it can be held naturally in the hand. The active pad surface is a 4" square that is slightly recessed with two push buttons located above it. The Koala Painter menu, displayed on the screen, is divided into three sections: Commands, Brushes and Colour Palette. The user is able to use the menu to build up a picture, combining freehand drawings with the basic shapes included in the menu and with previously stored designs, before deciding on colouring. A brush option allows the user to vary the thickness of lines.

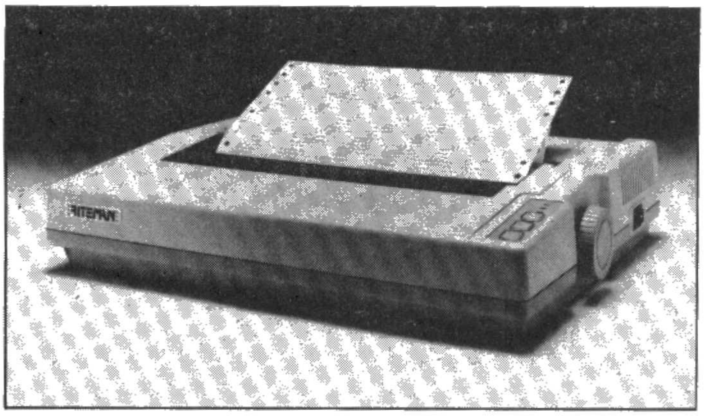
The command section of the menu gives the user total control over the formation of the picture. Freehand drawings can be supplemented by integral facilities for Lines, Frames, Boxes, Rays, Circles and Discs. The complete system costs £89.95 (including VAT). *Audiogenic Ltd., PO Box 88, Reading, Berks RG1 2SN.*



Hi-res printer

The Riteman A1 matrix printer from Micro Peripherals Ltd. is an 80 column printer weighing only 5kgs. The distributors claim superb print quality at a print speed of 120cps making it an ideal complement for personal and portable micro's.

Priced at £299 ex VAT the printer is fully featured with hi-res-graphics, italic printing and all the industry standard features with complete industry standard control code capability. *Micro Peripherals Ltd., 69 The Street, Basing, Basingstoke, Hants.*



NEW PRODUCTS

NEW PRODUCTS

Monitor

The TP200 from Philips is a compact monochrome video terminal which can be used both as an alphanumeric and graphic display. It incorporates a 12" high resolution CRT and can display up to 2,000 characters on the glare free P31 green phosphor screen.

The unit is mains powered and accepts a composite video signal so can be used with the majority of popular microcomputers.

The monitor was developed primarily for the home computer market and has been priced accordingly. It is said to give very good screen definition.

The new unit complements the Capital series of line monitors manufactured by Vako. These range in tube side from 3" to 12" monochrome to 14" colour. Vako also distribute a variety of optoelectronic products including Sharp LCD's and LED's and OKI plasma panels. *Vako Displays Ltd., Pass Street, Werneth, Oldham.*

Spectrum database

Following on the launch of Magpie for the Commodore 64, Audiogenic have announced Data Genie – a home database system for the 48K ZX Spectrum. Data Genie is an easy to use database and record retrieval package that allows Spectrum users to organise their own records in their own manner and to recall them under a wide range of parameters.

The package is controlled through the novel method of 'pop-up' menus which are managed by just three keys. The user selects the required option from each menu by raising or lowering a cursor line. Once the cursor is over the required option, a third key automatically pulls in the menu relating to that option, overlaying the new menu on the previous menu. The user is thus able to follow clearly the steps taken in building up the database.

As a menu element is selected, the software automatically writes the relevant part of the program. The user simply builds up his requirements from the menus and is unaware of the programming being undertaken. Once the set-up operations are completed, the user in effect has produced his own, tailor-made database program. Up to 146 records can be contained in memory at any one time. Each record can contain up to a maximum of 15 fields, ie 15 lines of data, each field consisting of between 10 and 20 characters.

Diskpen III

DISKPEN III, a new software program available from Henry's of Edgware Road, has been developed to provide an answer to all small-word processing requirements. When used with computers such as the Gemini Galaxy range, the Quantum 2000 range, the Kenilworth Personal and Nascom CP/M based units utilising the Gemini GM812 Intelligent Video Card, DISKPEN III is designed to offer a low-cost alternative to the larger, often under-utilised word processor.

Available in any format, the program includes a number of refinements which have been incorporated following research into user requirements and feedback from the previous Diskpen versions. For more specialist roles DISKPEN III can incorporate a number of optional extensions which provide an integrated approach to the needs of the small business, the laboratory and the home. The MAXiFILE extension for example allows DISKPENN III, to be employed as a free field database for a wide range of applications, including the keeping of stock, staff and home records and for use as a cross-reference index, day book and log book.

The SPooLER extension, meanwhile is useful for the production of a series of repetitive letters, allowing the printing of one document as the next is being prepared.

The third extension, MULTi-FORMAT has been developed for printing columns of text and can be used, for example, for the presentation of price lists. *Henry's, 404-406 Edgware Road, London W2 1ED.*

BBC Utilities Disc

Oakleaf Computers, have released a utility disc with a number of useful routines to format and verify discs and enable easy Tape to Disc transfers. The Formatter handles 40, 80 and 35 track discs and is unique in that it will not format discs with locked files until the files are released with the Acorn DFS * Access command. The catalogue of the disc is displayed prior to formatting, and the user has to confirm that the disc can be formatted. This feature almost entirely eliminates the risk of inadvertent destruction of expensive software. During formatting Track numbers are displayed in green, in the event of a Formatting error which is caused by a drive or interface error, the track number changes colour to blue, and the software attempts to reformat the offending track, in the event of further failure a diagnostic message is displayed. Once a track is successfully formatted, it is verified, if verification fails, perhaps due to a disc fault, the track number changes colour to red, and a repeat attempt is made, again after successive failures the diagnostic routine is entered, and the probable cause of the fault is displayed.

The diagnostic routines are included in both the Format and Verify utilities, and not only display a diagnosis of the fault, but suggest remedies that user can use to rectify the problem. As many of the Disc faults reported by users are easily rectified, this facility should be a real money, and time saver.

The other utilities on the disc ease the transfer of long files to disc. The command *MODBAS is used for Basic programs, and *MODMC is used for machine code files. Both these commands read in a file from tape or disc, and modify it to produce a new file which will both diable the DFS to prevent program corruption, and run the file from &E00. MODBAS, although in Basic can be handled exactly as Machine Code; that is *FILENAME will load and run the Basic File. This means that a BASIC file can be renamed as the Autoboot file !BOOT.

FSCONV changes the default Filing system selected for MODBAS or MODMC files to either ROM, NET, TAPE, or DISC.

The manual is priced at £15.00, and is zero rated for VAT. The associated software is free of charge, to purchasers of the manual. *Oakleaf Computers Ltd., Tel: 0476 76994.*

BBC Maintenance

In response to what will almost certainly become an increasing demand, a low cost on-site maintenance service has been introduced for the BBC micro range of equipment. The service is available through Acorn Dealers or direct from DDT Maintenance Ltd.

A £50 annual maintenance charge entitles BBC users to a one day engineer response whenever a fault arises. For an additional £25 a replacement machine will be provided if the repair must be completed in DDT's workshops.

NEWS

BBC second processor

Acorn's long awaited second processors for the BBC micro are still just that: long awaited; but Cambridge microprocessor Systems have come up with the goods in the form of a 6809 board with Tube interface priced at £249.

The new board is designed with industrial development systems in mind, but will no doubt find many applications in home computing. The board enables standard FLEX format discs to run on the BBC, supports high level language compilers, cross assemblers and disassemblers for most micros; it sits inside the BBC or plugs into an extension rack, and has a healthy 64K of dynamic RAM on board. Optional battery backup is available for CMOS devices, and the interface is the Acorn bus compatible DIN 41612.

The ability to operate FLEX will ensure a high degree of interest in this board. TSC FLEX is an increasingly popular 8K disc operating system. It is well documented and provides a powerful and comprehensive operating environment for software development. In this version a monitor ROM is supplied with the 2nd processor card and a BBC Basic program is supplied on disc for the BBC machine to link to the 2nd processor card. When linked together the system will accept a standard FLEX disc as a source to boot the FLEX system itself. FLEX supports a range of high level languages, including PL9, C, BCPL, Pascal, Fortran and FORTH.

Brother for the Brother

Last month *E&CM* reviewed the new Brother EP22 portable printer/typewriter. The machine was given a very favourable reception, with one or two reservations about the print quality and the editing facilities. We speculated on the possibility of a future machine on which these errors could be corrected; that machine has now become a reality in the form of the Brother EP44, due for launch within three months.

The initial specifications look very promising. The RS232 interface is to be retained, and so will the choice of thermal or carbon ribbon cassette printing, but on the new machine descenders will be available on a much improved print quality. The memory will be considerably larger than the 2K available on the EP22, and editing facilities will be extended. Perhaps the biggest bonus of all, however, is that the new machine will be able to send as well as receive information, enabling the Brother to be used as an extension to a microcomputer keyboard or as a terminal.

FORTH to BBC

Yet another product reviewed last month and now updated is Husband FORTH. The product, formerly available only on the ZX81, has now been implemented on the BBC micro and in a far superior form.

The new multitasking FORTH '83 does not restrict users to the single language as it does on the ZX81. The ROM fits into the BBC's sideways ROM board alongside the BASIC ROM, and is capable of handling disks and the usual files, thereby not upsetting operation. 16K of ROM is available, and none of the BBC's RAM is taken up.

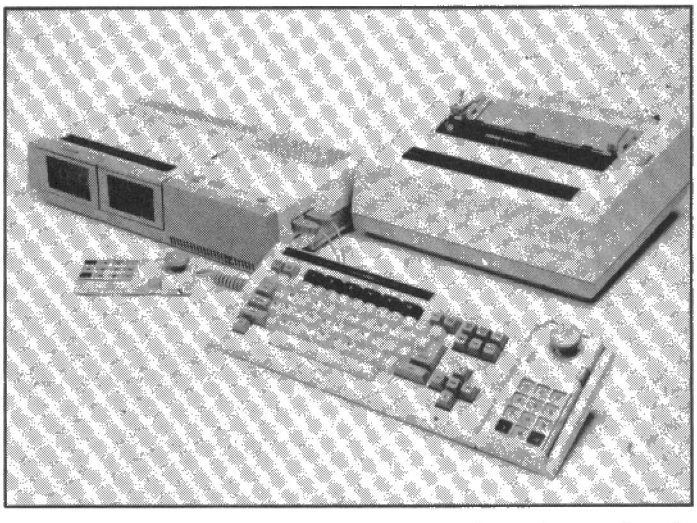
The author makes the proud boast that his version is twice the size and 'twenty times better' than other versions on the market. Certainly the multitasking capabilities should open up many new avenues for exploration. No price has yet been fixed for the chip (which like the ZX81 version is supplied with very extensive documentation) but it should be on sale by the time this magazine hits the news-stands.

Oric swallowed

Oric have now succeeded in their bid to be taken over by Edenspring (see *E&CM* December '83) and reap the resulting millions in investment (£4 million to be precise).

The money will be used to establish Oric in what they term the 'volatile home computer market': in particular, by developing a large R and D centre in Cambridge (where else?), and to finance a substantial advertising campaign in the press and TV. Further research will be made into the possibility of expansion into business communications and opto electronic systems, and the company is also co-operating with Inmos, the Government owned semiconductor manufacturer, on its future products.

120,000 16K and 48K Orics have now been shipped, and the company expects a first year turnover of £10 million: perhaps we can expect a new machine in the near future with all this money floating about?



The UK launch of CBS Colecovision's computer system — the Adam — is now confirmed for Spring 1984. The machine, pictured above, has already achieved a great deal of success in the states, based on promotion as 'the first complete home computer system for all the family' (quite a mouthful). The package is split into three component parts. Firstly, an 80K RAM console with digital data drive and inbuilt word processor; secondly, an 89 key sculptured keyboard of true professional quality; and thirdly, a letter quality printer capable of speeds of 120 words per minute. The Adam combines large memory capacity and word processing capability with a strong emphasis on games playing, built upon the success of Colecovision's video console, who's expansion modules plug into the Adam. The system is yet to be priced for the UK market, so we have yet to find out if the attractive looks are going to be worth paying for.

No problem

The computer press has recently been full of reports of a scarcity of add-ons for the Electron: it is not only Acorn who are having delivery problems. Promises of printer and joystick interfaces, ROM extension boards and Teletext adaptors from certain manufacturers have failed to materialise — certainly for the moment. This should however present no problem for *E&CM* readers. The magazine has already carried projects for the construction of an A/D converter and for an I/O port for the Electron, and this month readers will find a project for an RS423 interface.

Amstrad micro

Amstrad will soon be known for something other than the cheapest Hi-Fi stacks around. In April the company is to launch a home computer, but details of any kind of specification are being kept a close secret (assuming none of the usual 'inspired leaks').

A tight lipped Amstrad spokesman would say only that the company was 'looking into' and when pressed, 'working on' a computer 'for personal use', that is a home computer.

Two realistic conclusions can be drawn from this statement: firstly, that the machine will be in the Amstrad tradition of really cut-price goods; and secondly, that the 'looking into' process is fairly well advanced if an April launch date is to be met.

Updata

The 1984 Maplin buyers guide is now available, and as usual it offers an encyclopaedic range of electrical and electronics components, as well as all kinds of home computer hardware and software. The guide is priced at £1.35, and is available from Maplin Electronic Supplies Ltd., PO Box 3, Rayleigh, Essex SS6 8LR.

The 1984 edition of the 'Hobby Herald', BICC-Vero's product guide for the hobbyist, is now available. Over 100 new products are listed, including connectors for all applications and for microcomputers, telephone connectors, etching kits, enclosures etc., etc. The Herald is available for 50p from BICC-Vero Electronics Ltd., Retail Dept., Industrial Estate, Chandlers Ford, Hants SO5 3ZR.

Educational scientific suppliers Griffin & George have marked a bullish entry into the home computer with their first computer systems handbook. This contains information on a wealth of accessories, including disk drives, expansion units, interfaces, processors, application modules, robots, videos etc. etc., as well as the computers themselves. The catalogue is available from Griffin & George, Ealing Road, Alpperton, Wembley, Middlesex HA0 1HJ.

Paged ROM for the BBC micro

In recent months ROM based software has proliferated, highlighing the lack of spare paged ROM sockets on the BBC micro. In this article Peter Simpson and Brian Alderwick describe an extension board which overcomes the problem.

One of the most useful features on the BBC micro is its facility to store several languages or applications programs such as word processors in read only memory (ROM), ready for instant access via operating system commands. These ROMs are known as the 'paged' or 'sideways' ROMs. This is because the ROMs all appear in the same place in the memory map but with only one ROM being active at any one time. As supplied, the BBC micro has four paged ROM sockets, one of which contains the standard BBC BASIC interpreter, leaving three available for other uses. Disc users will find that one socket is taken up by the Disc Filing System leaving them with only two spare sockets.

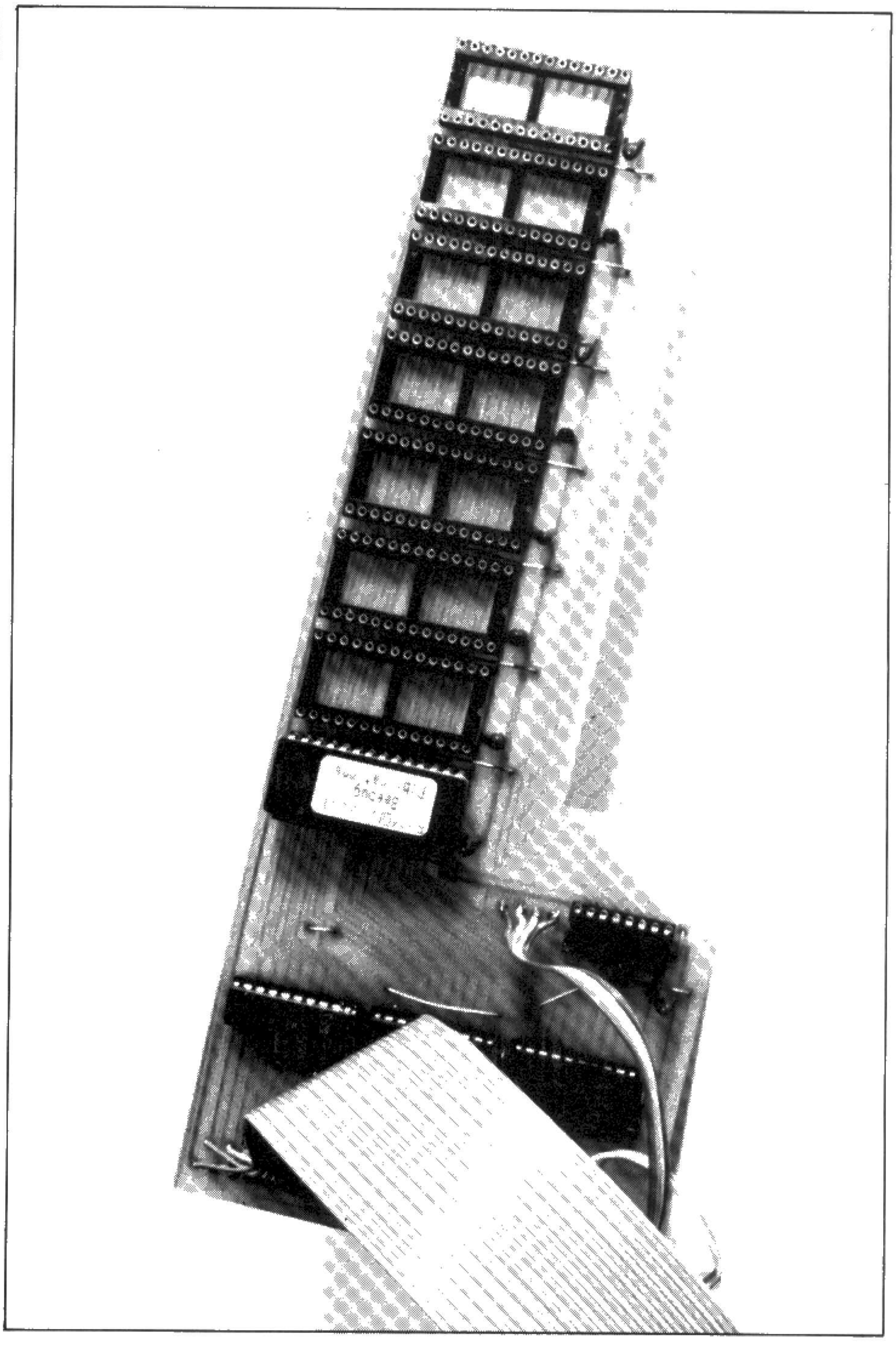
This article describes a board which provides seven additional paged ROM sockets, enabling a total of eleven paged ROMs to be present simultaneously within the BBC micro. A useful utility program is also included which prints out the names of all paged ROMs present in a BBC micro.

"the mysterious hole in the BBC keyboard is only half the story . . ."

The 1.0 and 1.2 operating systems are actually capable of supporting a maximum of sixteen paged ROMs, however, the design objectives of this project were to produce a simple, cheap and easily built single sided board and these dictated the final design and restricted the total number of sockets to eleven.

The ROM filing system

A little publisiced feature of the BBC micro is the ROM filing system, which is entered with the *ROM command. Most people associate this filing system with the speech processor and the 'mysterious hole' at the left hand side of the keyboard. In fact this is only half of the story and it is very surprising that Acorn have not made more of the



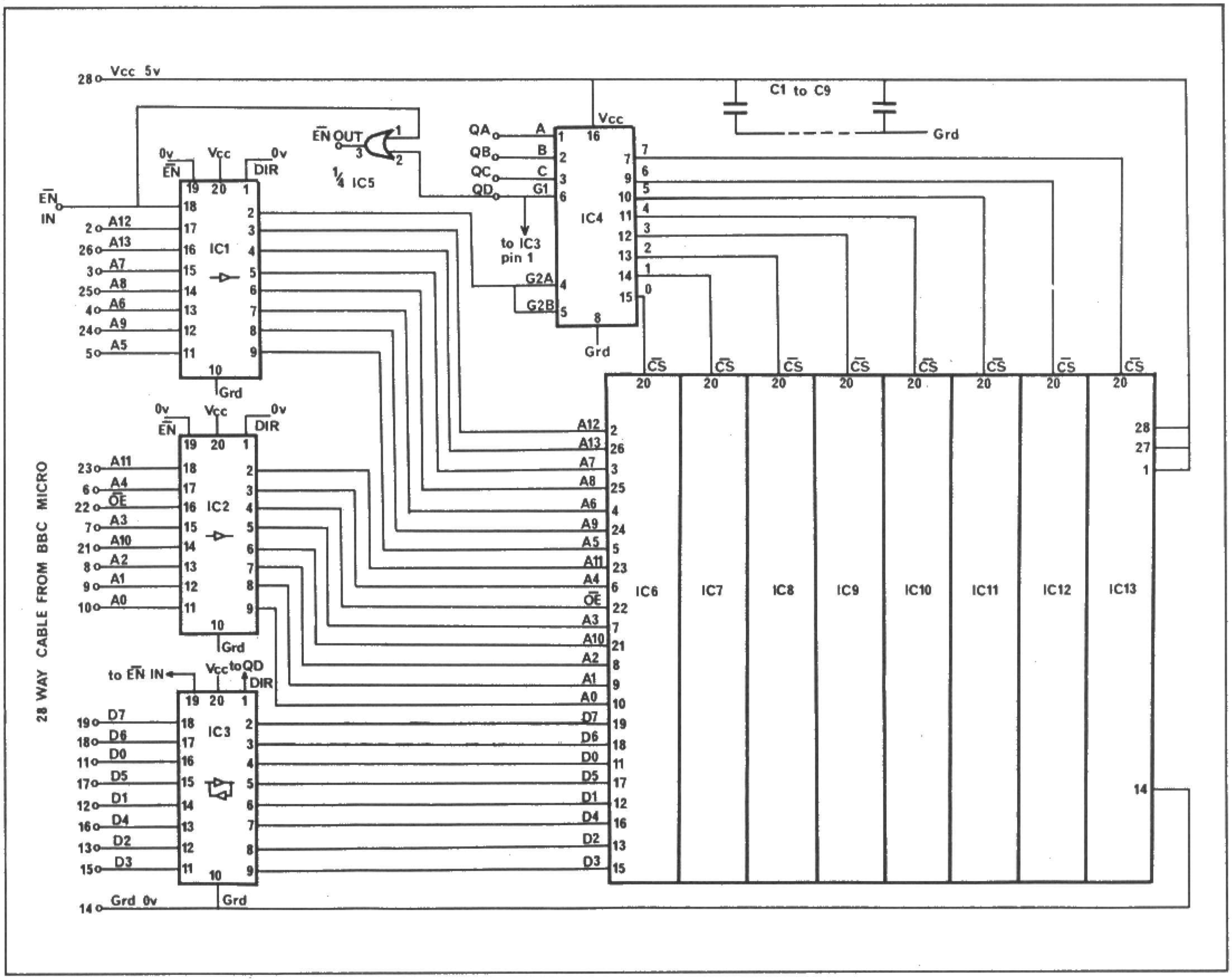


Figure 1. Circuit diagram.

> **"using the ROM filing system a 12K program loads in eleven seconds!"**

fact that ROMs can also be accessed from this filing system, although the ROMs have to follow a different protocol to the 'standard' paged ROMs. The ROM filing system has many similarities with the TAPE filing system including the *CAT command and the storing of programs and data in blocks. There are two major differences between the systems however, the first is that it is impossible to save or write to the ROM filing system (these give Bad Command errors) and the second is that the ROM filing system is considerably quicker than the TAPE filing system. For instance a 12k program loads in eleven seconds! Furthermore, data files as well as program files can be stored in the ROM filing system, for reading only of course.

This project allows up to ten of these ROM filing system paged ROMs to be present simultaneously giving almost instant access to any part of about 160k of program retained within the micro.

Acorn publish a technical note describing the ROM filing system and giving a program for creating a data file in the correct format, containing any mixture of BASIC or machine code programs or data files, ready for transferring to EPROM via an EPROM 'blower'. This is entitled: 'BBC Microcomputer ROM filing system'. A similar note is also available describing the sideways ROM protocol. These are priced at two pounds fifty pence each and are available from the Customer Services Department of Acorn Computers at Fulbourn Road, Cherry Hinton, Cambridge, CB1 4JN. The design for a suitable EPROM blower was published in the October and November 1983 editions of this publication.

ROM system operation

Which paged ROM is active, out of several present in the BBC micro at any time is determined, via intermediate hardware, by the contents of the 'paging register'. The paging register is located by the BBC micro's internal address decoding at address &FE30. The chip is a 74LS163 which is a four bit presettable counter/divider. Acorn do not use the counting/dividing facility, being content simply to preset the counter outputs with the required paged ROM number. The 'presetting' inputs are connected to the lower four bits of the data bus (D0 to D3) and the 'load' input is connected to the decoded address line corresponding to &FE30. Thus to preset the outputs it is only necessary to load the accumulator with the paged ROM number, in the lower four bits, and to write this to the address &FE30 using, for example, the assembly language instruction STA &FE30. After this instruction the outputs will be set to the states held by the lower four bits of the data bus and will retain these values until either the machine is switched off, or a new STA &FE30 instruction is executed. The value represented by the four bits is now said to be latched, since the data bus can change to any new value without affecting the 'contents' of the latch. Four binary bits can represent sixteen individual numbers from

0 to 15, and this is the origin of the BBC micro's limit of a maximum of sixteen paged ROM's.

The BBC circuit board itself only has space for four paged ROMs and the decoding hardware included on the board reflects this situation. To enable any particular paged ROM, its 'chip select' pin (pin 20) must be held low. The paged ROM sockets on the BBC board are each enabled by one output from a two to four decoder IC, a 74LS139. This chip accepts a two bit binary number, which can take one of four possible values (0 to 3), and pulls one of its four outputs low for each input value. The chip's two input lines are connected to the two least significant output lines of the 74LS163 latch. Thus the link is completed between the value held by the accumulator during the STA &FE30 instruction and the chip enable input of the particular ROM or EPROM which is being selected.

Unfortunately this is not the whole story, if it were, the BBC micro would not work at all! As described the selected paged ROM would be enabled at all times, even when the microprocessor was trying to access the RAM memory, which lies from address 0 to &7FFF, or the operating system ROM which lies from &C000 to &FFFF. The part of the memory map which is set aside for paged ROMs is the address range &8000 to &BFFF and it is important that the selected ROM is only enabled when reading from an address within this range. Two NAND gates, within the BBC micro, synthesise a suitable enable signal from the two highest address lines A14 and 15 and this is connected to the enable input of the

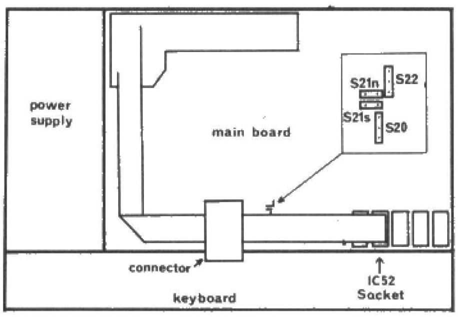


Figure 4. Position of ROM board inside micro case.

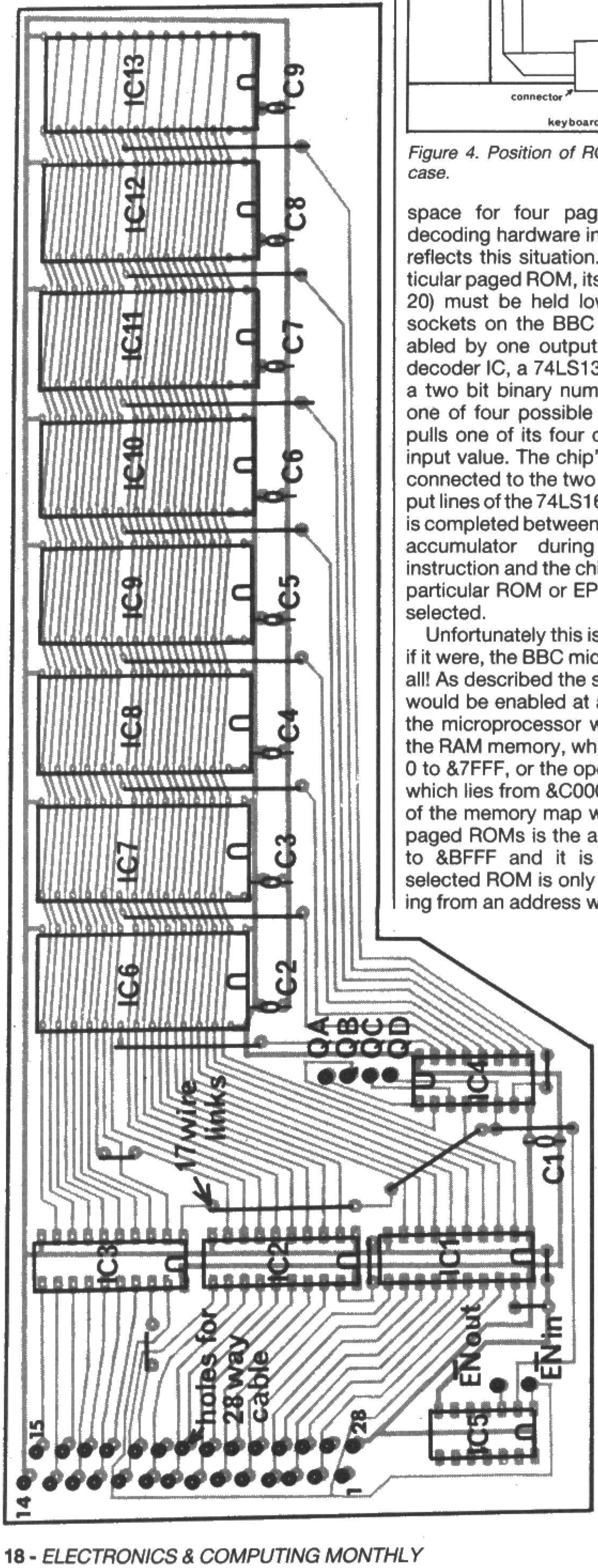


Figure 3. Overlay of components.

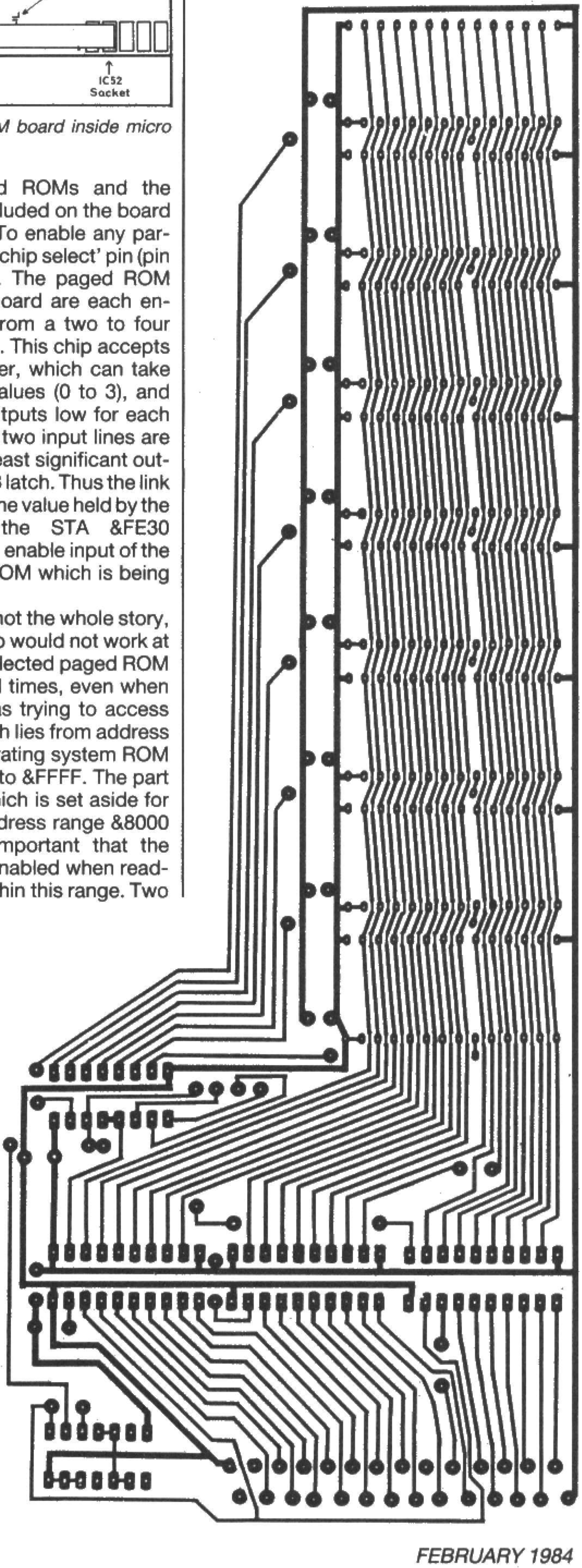


Figure 2. Foil layout.

74LS139 two-to-four line decoder. Thus if the address being accessed is within the paged ROM area, then the decoded output line is allowed to go low, enabling the chip. However if the address is not within the paged ROM area, then the decoded output is forced high, even though the two input lines from the latch say that it should go low, thus disabling the paged ROM.

> **"each paged ROM socket on the BBC is not one socket but four . . ."**

The two least significant bits of the latched paged ROM number are connected as previously described to the inputs of the two-to-four decoder. But what has happened to the other two bits? The answer is not a lot! In fact they are left floating and are not connected to anything. The consequence of this is that each paged ROM socket in the BBC micro is not just one socket but four sockets! Socket number zero for instance is also socket number four, socket number eight and socket number twelve! The reason for this can be seen by writing down the binary numbers from 0 to 15, ie. 0000 to 1111. If the two most significant (the two leftmost) bits are covered with a sheet of paper, then a pattern consisting of the values 00, 01, 10 and 11 will be seen to repeat itself four times in the list. These are equivalent to the two bits of the latch which are transferred to the decoder and it is clear that the state of the upper two bits is irrelevant in determining which socket is selected. Thus a socket can be selected by any one of four different numbers, just because these numbers happen to share the same pattern of bits in the two least significant positions. Clearly there is only one real socket with one real number, the other numbers which access the real socket are said to access 'images' of it.

The problem is now reduced to, 'Spot the real socket!' Acorn have come to the rescue here by arbitrarily defining the real socket number to be the highest socket number which will access the socket. Thus in the example just given, where the numbers were zero, four, eight and twelve, the socket would be numbered twelve with zero, four and eight being the images. The 1.2 operating system contains some sophisticated software to distinguish the images from the real sockets. This software compares the contents of any paged ROM socket, with those of all the other paged ROM, sockets. If two sockets are found to have identical contents then the lower numbered one will be ignored, thus ensuring that only the highest socket numbers are accepted. The real socket numbers for the four sockets in the BBC micro are thus twelve, thirteen, fourteen and fifteen.

Circuit Description

The circuit diagram is shown in **Figure 1.** All address and data lines, together with the output enable signal OE, are buffered by 74LS245 bidirectional buffers, IC1 to IC3. In this application all the signals are unidirectional, but these chips were used because of their convenient pin layout compared to their unidirectional counterpart. After passing through the buffers the address lines A0 to A13, the data lines D0 to D7 and the output enable signal pass directly to the eight paged ROM sockets IC6 to IC13 whuch are connected in parallel.

The chip select signals are generated by IC4, a three-to-eight line decoder, whose outputs are connected, one each, to the paged ROM sockets. These signals are active low, as required by the ROMs, which means that the ROM is disabled when the line is high and enabled when the line is low. Pin one of the paged ROM sockets is connected to 5V as required by the manufacturers of 2764 and 27128 EPROMs, which is contrary to the situation on the BBC circuit board where this pin is not connected to anything. This allows EPROMs from all manufacturers to be used on the extension board with impunity.

The four bit value stored in the paging register is transferred to the extension board via four wires soldered to the register chip. Of these signals (Qa to Qd), Qa to Qc go directly to the three to eight line decoder, whilst Qd is used to enable the decoder and the data bus buffer when appropriate.

Hardware construction

A complete parts list is given below. The ROM card is made of single sided printed circuit board. This was chosen to ease home construction but necessarily contains more links than would be the case with double sided construction. A foil layout is given in **Figure 2.** As can be seen, the pads on the 28-pin ROM sockets are very small and it is necessary to use a 0.7mm dia. drill to leave enough land to solder to. If this task seems rather daunting, a board is available through the *E&CM* PCB Service. A very fine point soldering iron should be used during construction and great care taken not to bridge across the fine tracks. It is best to mount all ICs in sockets, which helps if subsequent troubleshooting is necessary.

It should be noted that there are many kinds of sockets on the market and in the case of the 28 pin ones which contain the ROMs, the better the quality of the socket the easier it will be to fit and remove the paged ROMs. Note that IC4 pin 1 faces north whilst all other ICs face south. A component overlay appears in **Figure 3.**

The board is connected to the micro by a 28-way ribbon cable, a two-way cable and four-way cable. The 28-way cable is soldered to pads on the board, the other end being terminated in a 28-way DIL header which can be a solder type or cut down 40-way Insulation Displacement Connector. This latter kind is not, unfortunately, available in a 28-way version. Make sure that pin 1 on the header is connected to pad 1 on the board etc. The two-way cable is terminated with two small crimp-on terminals which are pushed onto the two pins exposed when link S21 south is removed. This link is located on the main BBC board between IC21 and IC22 and is the one lying in an east-west direction closest to the keyboard. The east pin is connected to EN IN on the ROM board and the west one is connected to EN OUT. The four-way cable connects points Qa, Qb, Qc and Qd on the ROM board to pins 14, 13, 12 and 11 respectively of IC76 which is located near the keyboard connector on the main BBC board. If your particular micro has this IC soldered in, it will be necessary to solder the wires carefully to the pins. If it is socketed, then it is suggested that a spare is purchased and the wires soldered to this, leaving the original IC available for re-insertion if the ROM board is removed.

The board is fitted to the micro as shown in **Figure 4.** The ribbon cable should be neatly bent through 90 degrees, passed between the keyboard and keyboard connector and plugged into socket IC52. The mounting of the board in the computer is left to the constructor and depends on what materials are available and whether he/she is willing to drill the case. The board is physically compatible with the Sideways RAM board published in the December 1983 and January 1984 issues of this publication and can be mounted above it on small stand-off supports. ■

PARTS LIST

TTL Integrated Circuits	
IC1-3	74LS245
IC4	74LS138
IC5	74LS32
ROM or EPROM	
IC6-13	2764, 27128 or ROM
Sockets	
14-pin DIL	For IC5
16-pin DIL	For IC4
20-pin DIL	For IC1-3
28-pin DIL	For IC6-13
Capacitors	
C1-9	100n
Printed Circuit Board	
Single Sided	94mm x 240mm
Plug	
28 pin DIL Header	(see text)
Crimp on terminal	(to fit Molex connector S21 on BBC main board

Cable

28 way (cut down 34 way) ribbon x 500mm cut to suitable length. 4 way ribbon x 180mm. 2 way ribbon x 180mm. Wire for links.

Next month: operation of the extension board, protocols, and 'Super HELP' software.

MICRONET
a hackers' guide

Mike Brown* has the low down on micronet's communications protocols.

Micronet 800 makes use of BT's Prestel system to provide micro users with a 24 hour a day telesoftware service. Prestel manages to provide a good telesoftware medium capable of providing a wide range of information with a high degree of accuracy. It manages to do this by way of some cunningly concealed software within the prestel frames which is even more cunningly used by Micronet terminals.

Before revealing the secrets of Micronet though, it would be a good idea to take a look at the general concept of Prestel.

Into Prestel

Prestel was a disarmingly simple idea: to use modified TV sets as terminals to a network of user-friendly computers for the purpose of information retrieval. This simplicity, for good and ill, has pervaded Prestel's development throughout its short history.

Information Providers (IPs) are organisations which supply and amend information on the system for subscribers to access. Subscribers pay (in addition to telephone costs) a standing charge plus a connect time charge during office hours. IPs must pay a service charge of £6000 per annum and an annual charge proportional to the amount of information they provide as well as any costs they may incur as users. IPs are allowed to charge users if they wish for access to parts of their database.

The behaviour of a Prestel terminal is defined in the PRESTEL TERMINAL SPECIFICATION (available from Prestel). Its major sections refer to modem and line characteristics and character format and coding. Prestel sets are allowed to have auto-diallers so long as they conform to the loop disconnect dialling principles of the public telephone network. The Prestel modem is a conventional V24 modem set to receive data at 1200 baud and transmit at 75 baud in full duplex mode. Character coding is seven bits data followed by one bit even parity parenthesised by one start and one stop bit.

Display Format

The Prestel terminal screen consists of 24 lines each of 40 characters. Displayable characters appear upon reception at the position of the cursor and the cursor is advanced along the line; at the end of each line the cursor advances to the start of the line below; at the end of the screen the cursor advances to the first character position of the top line. In short, it is a paged, rather than scrolling, screen: Prestel will normally send CURSOR HOME, to move the cursor to the top left of the screen, followed by 23 lines of data. Lines of less than 38 characters are terminated by carriage-return-line-feed (CRLF); lines of 38 or 39 characters are padded with spaces to 40 characters; lines of 40 characters are not terminated —

*Technical Manager MICRONET 800.

BITS b7 b6 b5 b4 b3 b2 b1	b7 b6 b5	0 0 0	0 0 1	0 1 0		0 1 1			1 0 0		1 0 1		1 1 0		1 1 1	
	Row \ Col	0	1	2	2a	3	3a	3b	4	4b	5	5b	6	6a	7	7a
0 0 0 0	0	NUL		Sp		0	🬏		@		P		–	🬞	p	🬭
0 0 0 1	1		Cursor on	!	🬀	1	🬐	Set verify mode (1)	A	Alphanumeric Red	Q	Mosaic Red	a	🬟	q	🬮
0 0 1 0	2			"	🬁	2	🬑	Set verify mode (2)	B	Alphanumeric Green	R	Mosaic Green	b	🬠	r	🬯
0 0 1 1	3			£	🬂	3	🬒	Skip block	C	Alphanumeric Yellow	S	Mosaic Yellow	c	🬡	s	🬰
0 1 0 0	4		Cursor off	$	🬃	4	🬓	Set programme mode	D	Alphanumeric Blue	T	Mosaic Blue	d	🬢	t	🬱
0 1 0 1	5	ENQ		%	🬄	5	▌		E	Alphanumeric Magenta	U	Mosaic Magenta	e	🬣	u	🬲
0 1 1 0	6			&	🬅	6	🬔		F	Alphanumeric Cyan	V	Mosaic Cyan	f	🬤	v	🬳
0 1 1 1	7			'	🬆	7	🬕		G	Alphanumeric White	W	Mosaic White	g	🬥	w	🬴
1 0 0 0	8	Active Position Backward (APB)		(	🬇	8	🬖		H	Flash	X	Conceal Display	h	🬦	x	🬵
1 0 0 1	9	Active Position Forward (APF)		)	🬈	9	🬗		I	Steady	Y	Contiguous Mosaics	i	🬧	y	🬶
1 0 1 0	10	Active Position Down (APD)		.	🬉	:	🬘		J		Z	Separated Mosaics	j	▐	z	🬷
1 0 1 1	11	Active Position Up (APU)	ESC	+	🬊	;	🬙		K		←		k	🬨	¼	🬸
1 1 0 0	12	Clear Screen (CS)		,	🬋	<	🬚		L	Normal Height	½	Black Background	l	🬩	‖	🬹
1 1 0 1	13	Active Position Return (APR)		–	🬌	=	🬛		M	Double Height	→	New Background	m	🬪	¾	🬺
1 1 1 0	14		Active Position Home (APH)	.	🬍	>	🬜		N		↑	Hold Mosaics	n	🬫	–	🬻
1 1 1 1	15			/	🬎	?	🬝		O		#	Release Mosaics	o	🬬	■	█

Figure 1. The Prestel transmission codes showing the alphanumeric set (columns 2 to 7), the alphamosaic set (columns 2a, 3a, 6a and 7a), the display modifiers (columns 4b and 5b) and the cursor control characters (columns 0 and 1).

Attribute	Display Modifier			Attribute	Display Modifier		
Alphanumeric display	ESC A	65	red		ESC S	83	yellow
	ESC B	66	green		ESC T	84	blue
	ESC C	67	yellow		ESC U	85	magenta
	ESC D	68	blue		ESC V	86	cyan
	ESC E	69	magenta		ESC W	87	white
	ESC F	70	cyan				
	ESC G	71	white	Conceal display	ESC X	88	
Flashing	ESC H	72		Continuous alphamosaics	ESC Y	89	
Steady	ESC I	73		Separated alphamosaics	ESC Z	90	
Normal height characters	ESC L	76		Black background	ESC ½	92	
Double height characters	ESC M	77		New background	ESC →	93	
Alphamosaic display	ESC Q	81	red	Hold mosaic	ESC ↑	94	
	ESC R	82	green	Release mosaic	ESC #	95	

Table 1. Prestel display attributes and their modifiers

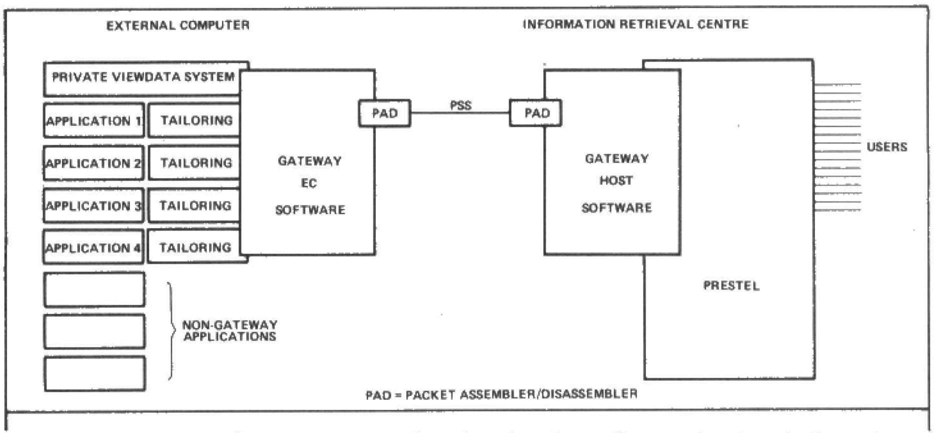


Figure 3. The Prestel Gateway connection showing the software structure in the external computer. Most external computers run a private viewdata system which is interfaced directly to the Gateway EC software. Applications designed for non-Prestel terminals must each be tailored for use over Gateway. External computers may also, of course, run applications which are not available to Gateway users.

the terminal is expected to reposition the cursor automatically.

The character format of Prestel terminals conforms to the Broadcast Teletext Specification as shown in **Figure 1.** In a 7-bit coding system 128 character codes are available designated 0 to 127 decimal (or 00 to 7F hex). The representation of codes 32 to 127 (20 to 7F hex) closely resembles the displayable characters of the International Alphabet No 5 (IA5) except that room is made on Prestel for both the £ and $ signs to be included. The codes 0 to 31 (00 to 1F hex) are used mainly for cursor control and are not displayed.

The ESCAPE character (code 27 or 1B hex) has special significance. When ESCAPE is received the terminal waits for the next character, the pair being known as a display modifier, ie. a command to the terminal to change some attribute of the display. The attributes and their display modifiers are are shown in **Table 1.**

For example, upon receiving ESC D the terminal will print subsequent chartacters in blue. At the beginning of every line the display will revert to steady normal height white alphanumerics on a black background regardless of modifiers which are in force on the previous line. Thus, to produce the word HELLO flashing in double height blue the sequence:

ESC H ESC M ESC D H E L L O

must be sent to the terminal.

After receiving a modifier ESC Q to ESC W subsequent displayable characters (ie those in Columns 2, 3, 4, 5, 6 and 7) appear as alphamosaics. Note that those characters in Columns 4 and 5 remain unaffected by these modifiers and are called "blast through" alphanumerics. Alphamosaics are used for building pictures and headings.

"...cunningly concealed software which is even more cunningly used by MICRONET terminals".

Prestel In Action

Prestel is a very straightforward information retrieval system. Users find the information they require through a succession of frames displayed on their screens choosing at each stage from lists of numbered options designed to refine gradually the field of interest. Searching for information is therefore very slow but has the virtue of simplicity. The browsing technique which this simplicity encourages is also useful because alternative avenues of exploration are continually suggested to the user. Keyword searching, often quoted as a more powerful alternative, offers the user little chance of meandering.

The Prestel network consists of one central Update Centre (UDC) called DUKE connected by high-speed data lines to six Information Retrieval Centres (IRCs) which supply Prestel's 35000 subscribers with information **(Figure 2).** Four IRCs are in London and two are in Birmingham. Each holds a database of 250,000 frames and can support 200 simultaneous users. IPs amend their information on DUKE using either a conventional Prestel terminal or by 'bulk-updating' from their own computer systems. DUKE immediately broadcasts any amendments made by IPs to all the IRCs.

Attached to each pair of IRCs are multiplexer lines to major cities. Local call access to Prestel is now available to 92% of the telephone population.

Simple messages can be exchanged between Prestel users via the MAILBOX service. Text is typed onto special frames and completed messages are sent to the destination storage area where they are queued to await inspection. This facility is only available on the Enterprise computer but by early next year messages may be sent and collected from any IRC even though Enterprise will probably still be used as the message store.

Prestel terminals may also be used to communicate, via Prestel, to private external computer systems. This avoids the user having to dial the external computer directly from a different type of terminal and has the advantage of local call access no matter where the external computer is situated. This so-called "Gateway" connection is effected by the allocation of a number of virtual circuits onto PSS from the IRCs to the external computer **Figure 3.**

Many external computers have databases which can be searched in an identical manner to Prestel. Such databases can be much larger than Prestel's and greater sophistication of search technique is often employed. External computers (ECs) are particularly

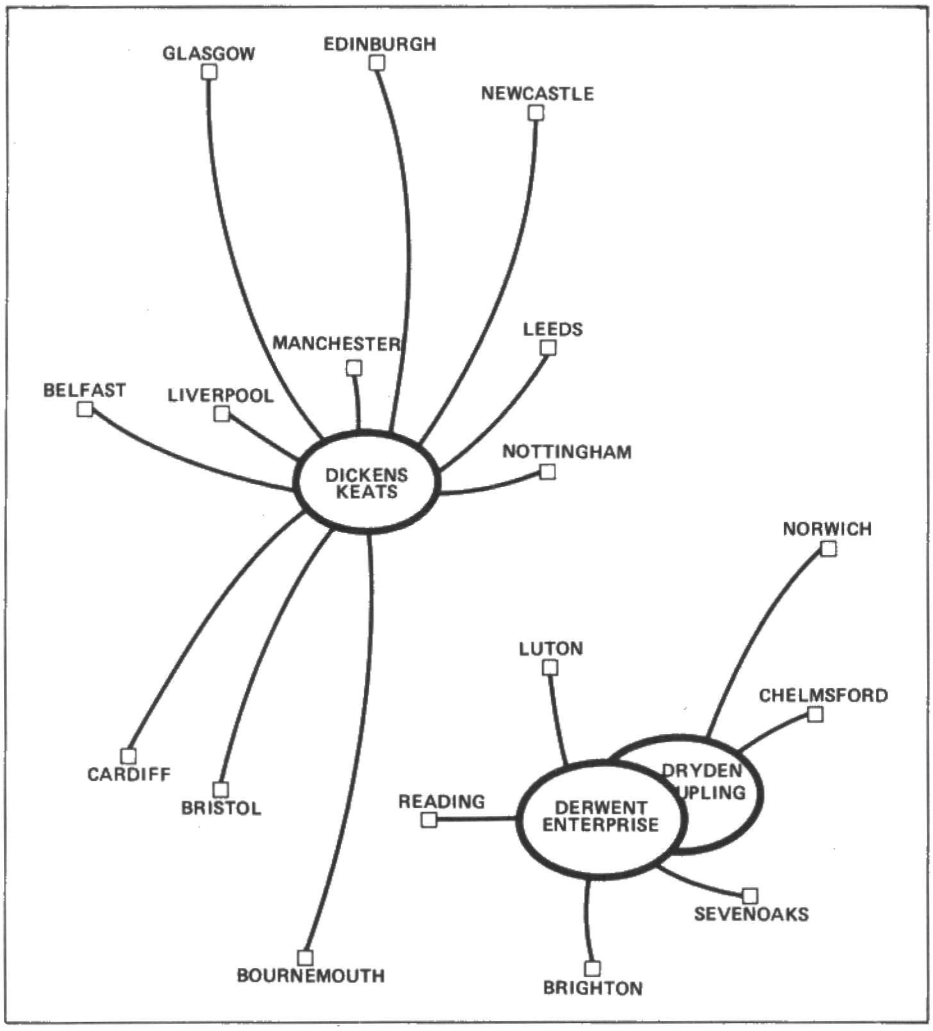


Figure 2. The current Prestel topology showing the two Birmingham and four London IRCs and the network of multiplexer links which will extend local call access to 92% of the UK telephone population.

useful for truly interactive applications (eg flight booking and electronic blanking) rather than just information retrieval. Each application on the EC must be tailored for Prestel terminals: at the very least the data to the terminal must contain only valid Prestel characters and be segmented into Prestel frames. It is usually wise, however, to make rather more fundamental changes to applications software to take account of the graphics and colour capabilities of Prestel terminals. Such tailoring, concerned as it is with issues of meaningful presentation of information, is labour-intensive and often costly.

Telesoftware

Building upon the pioneering work done by the Council for Educational Technology (CET) and Prestel's Aladdin's Cave, Micronet 800 was launched in March 1983. Micronet 800 is a subscription telesoftware and microcomputing news service run jointly by Prestel and Telemap Ltd. There are now over 6000 Micronet members using predominantly the BBC Microcomputer and Sincliar XZ Spectrum with some Commodore Pet, Apple II and Tandy TRS-80.

In 1979 CET began to develop software to allow a microcomputer to behave as a Prestel terminal. It was soon discovered that, with certain compromises on Prestel display, a microcomputer was more than capable of such a task with the help of an external modem.

At about the same time the BBC and IBA were looking into the feasibility of broadcasting software digitally over Ceefax and Oracle, the two teletext services, to purpose-built intelligent teletext terminals, a technique which came to be known as telesoftware. CET decided to investigate the same idea on Prestel but for transmission to popular microcomputers.

Prestel as a telesoft medium has many important advantages over teletext: Prestel is a massive information retrieval system available 24 hours a day which can store thousands of programs all of which are instantly accessible. Teletext may only broadcast a handful of programs and only while the host TV station is on the air.

Character coding and transmission errors are problems which are common to all data transfer systems. There are some character codes which are illegal within a Prestel frame; telephone line noise is no more than an inconvenience when it appears in text read by a human but is intolerable in a program delivery mechanism. To solve these problems CET developed a file transfer protocol which was published in 1981. It proved to be popular amongst those working in the field and became the de facto standard for all telesoftware on Prestel.

The CET document 'Format Recommendations for Prestel Telesoftware' describes the conversion of an 8-bit file to one consisting only of valid Prestel codes, its segmentation into frame-siezed blocks and the arrangement of these blocks on Prestel.

Conversion

In order to allow any 8-bit file to be stored and downloaded from Prestel frames a series of control sequences are used to shift those non-Prestel characters into the printable range (in a similar way to the alphamosaic modifiers). The terminal processes these control sequences upon reception to regenerate the original characters. For ordinary text files or source programs the overhead of this conversion is very small but for full 8-bit files the size may more than double **(Figure 4).**

Segmentation

Since Prestel frames can hold a maximum of 878 characters of IP data long files must be broken into a sequence of frames. This

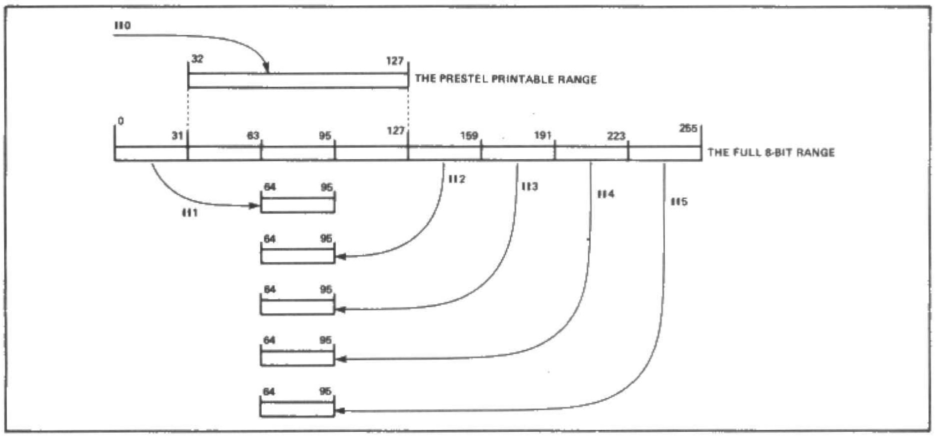
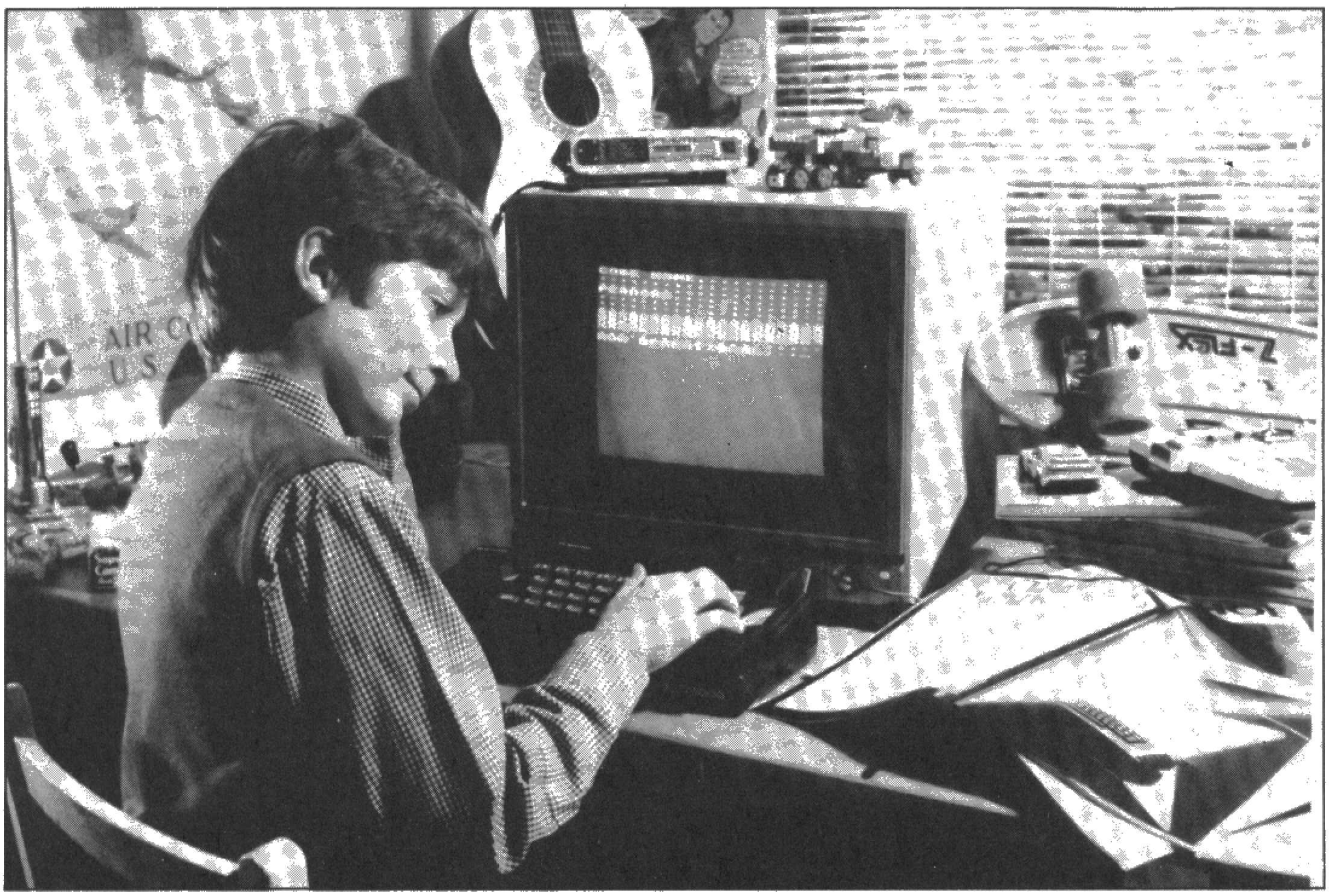


Figure 4. Diagram showing how characters from the full 8-bit range are mapped to characters safely within the Prestel printable range by the use of the locking shift control sequences ||1 to ||5. The "shift out" control sequence ||0 instructs the terminal to treat subsequent characters without mapping.



While not being able to start World War three via Micronet, an understanding of how the service operates enables all its features to be exploited.

segmentation provides a convenient basis for error protection beyond that of simple horizontal parity on each character. The block of characters on each frame is parenthesised by the control sequences 11A and 11Z and is followed by a vertical parity checksum which is the result of bitwise exclusive-OR calculations on every character in the block.

Upon receiving a telesoftware frame the terminal can compare the checksum which it has calculated with that received. Any discrepancy indicates that corruption has occurred during transmission and the frame must be re-requested. This, combined with the horizontal parity check, ensures a very low probability of data errors passing undetected.

Arrangement of Blocks

The blocks of a telesoftware file are arranged on consecutive frames of a Prestel page (spilling onto further pages if necessary) preceded by a special header frame which contains information about the name of the file and its length in blocks. Although the maximum number of blocks is 999 files are rarely longer than 20 and almost never over 50 blocks.

Beyond CET

The CET protocol approaches telesoftware in the context only of the delivery of files to the backing storage of a terminal but there has been a recent trend toward "load and go" telesoftware, that is, where programs are delivered into the terminal's RAM for immediate execution. The introduction of ROM based terminal software liberates most of the computer's user RAM for this purpose so that the delivery mechanism presents little restriction on size of program. Not only is load-and-go software more attractive to the user but it also allows the terminal to be reconfigured for interactive on-line applications.

One problem which arises is one of developing a format and downloading strategy which allows complex multi-file software to be downloaded as a single user-operation for immediate execution from RAM but also allows faithful permanent storage on the user's micro. Scicon Consultancy International Ltd. has designed such a format for Micronet 800 which lies within the CET recommendations making the downloading of modern commercial software very simple and effective.

The committee which polices and defines the CET format met recently to discuss a number of issues of its clarification and possible extension. The committee emphasised the importance of the horizontal parity check. If this is not done it can prejudice the effectiveness of the vertical parity check and if line noise is present always increases the time taken to download a file. Amongst the suggestions for extension of the format were:

1 An optional enhancement to the sequencing information on each block allowing multiple blocks to appear on a single frame. This is for specialist use only when it is necessary to download small items of information in application such as flight booking.

2 The addition of two new optional control sequences for use when downloading software from external computers via Gateway which partition title information and data.

Extensions to Prestel

Prestel are looking into the general issue of communication between an external computer and an intelligent Prestel terminal. Afraid that without some direction IPs may develop communication with such terminals in an amorphous way, Prestel are considering a layered approach based very loosely upon the International Standards Organisation (ISO) seven-level model for open systems interconnection (OSI). Unlike OSI, however, the Prestel version involves a distinctly master-slave relationship between external computer and terminal. The Physical level is already defined elsewhere and the Session and Data Transfer levels are currently under discussion. The terminal is viewed as a device which minimally behaves as a Prestel terminal but which may be capable of running a number of concurrent processes which consume data from and send data to the external computer. ■

Secrets of the Spectrum's

Mike James reports that use of the Spectrum's channels and strea

If you have any special I/O device connected to your Spectrum or if you are planning to build any new devices then the problem of how to send data to it or receive data from it will have occurred to you. It is usually thought to be easier to construct a hardware interface to the Spectrum than a 'software interface' with ZX BASIC. However the Spectrum has a very flexible and sophisticated method of handling I/O based on the use of 'streams' and 'channels' that makes it comparatively easy to interface special I/O devices to ZX BASIC in a way that allows them to be treated on a par with Sinclair's own hardware. Using some very simple Z80 assembly language software it is possible to send data to any device using PRINT and LIST and receive data from any device using INPUT and INKEY$ and this means that applications programs can be written using standard ZX BASIC without lots of machine code USR function calls.

Before getting on to more technical topics it is worth giving a brief outline of the ZX BASIC commands that are relevant to stream and channel I/O. It is surprising that although the standard Spectrum uses such a flexible I/O technique there is hardly any mention of it in the manual!

Channels and streams

A good way of thinking about I/O is to separate it into two parts, one corresponding to the software that receives or generates the data and the other corresponding to the hardware that receives or generates the data. In ZX BASIC the software component of I/O is referred to as a 'stream' and the hardware component is referred to as a 'channel'. The key difference is that a stream is a featureless flow of data into or out of a program but a channel corresponds to a particular I/O device such as the ZX Printer. Think of a stream as a collection of data items on their way to or from some piece of hardware.

Stream I/O commands

A stream is identified by a number in the range 0 to 15 and their basic operations are reading and writing data. The instruction:

```
INPUT $s;'input list'
```

will read data from stream 's' into the variables in the 'input list'. For example:

```
INPUT $0;A;B;A$
```

will read data from stream 0 and store it in the variables A, B and A$.

In the same way the command:

```
PRINT $s;'print list'
```

will send data to the stream 's' from the variables in the 'print list'. For example:

```
PRINT $0;TOTAL;A$
```

will send data to stream 0 from the variables TOTAL and A$.

Notice that both the INPUT $ and PRINT $ can be used in exactly the same way as the ordinary INPUT and PRINT statements. Any item that you can use as part of any normal 'input list' or 'print list' can be included as part of the stream I/O statements. For example:

```
PRINT $2;"HI THERE";TAB(10);"FOLKS"
and
INPUT $0;"What is your name";N$
```

are both valid. The PRINT statement sends the literal string "HI THERE" then ten spaces (blanks) followed by the literal string "FOLKS" to stream 2. Notice that the data that is sent to the stream is exactly the same as the data that would be sent to the screen. The INPUT statement is a little more complicated in that it not only requests data from stream 0 it also sends data in the form of the literal string "What is your name". Each stream number is in fact associated with two streams of data — an input stream and an output stream. Data written to the stream by either PRINT or INPUT is sent to the output part of the stream and any data read from the stream is obtained from the input part of the stream.

The only other two stream I/O commands that can be used with the unexpanded Spectrum are LIST and INKEY$. The full form of the list command is:

```
LIST $s,n
```

where 's' is the stream number that the program is to be listed to and 'n' is the line number that the listing will start from. For example:

```
LIST $1
```

will list an entire program to stream 1.

The other stream-oriented command, INKEY$ can be used to return a single character (byte) from any stream that is associated with a device that supports input. The function:

```
INKEY$ $s
```

will return a single character from the stream 's'. If no character is available from the input device then the null string is returned.

The channel commands

The idea of a stream of data is easy enough to understand but you might be wondering how the stream numbers are associated with hardware I/O devices? The answer is that before any data is sent or received over a stream it has to be OPENed. OPENing a stream serves two purposes, it associates a stream number with a particular I/O device and signals the I/O device that it is going to be used. Often as well as just signalling that a device is about to be used OPENing a stream involves initialising the device to get it into a state where it can be used, however, such initialisation depends very much on the device itself. To OPEN a stream ZX BASIC provides the commands:

```
OPEN $s,c
```

where 's' is the stream number being opened and 'c' is a string specifying the channel that it is being associated with. Following this command the destination of any data sent to the stream 's' will be the channel 'c', which will also be the source of any data read from the stream. Before a practical example of using the OPEN command can be given we need to know what channels the Spectrum has.

"... allows I/O devices to be treated on a par with Sinclair's own hardware".

The unexpanded Spectrum (ie without Microdrives) recognises only three different channels:

K — the keyboard channel
S — the screen channel
and P — the printer channel

"... the manual makes lit
tech

Thus,

```
OPEN $5,"K"
```

OPENs stream 5 and associates it with the keyboard. Following this command

```
INPUT $5;A;B
```

will get data from the keyboard in the same way that a normal INPUT command would. However, the command

```
PRINT $5;"HI THERE"
```

now sends data to the output side of stream 5 which is associated with the keyboard's display area at the bottom of the screen. Thus the literal string "HI THERE" is printed in the lower part of the

EXCLUSIVE *

streams and channels

s enables a range of I/O devices to be easily interfaced to the computer.

screen normally reserved for INPUT messages. If you try this you are unlikely to be able to see the string as the lower area of the screen is cleared when a program halts or when an INPUT statement is encountered. If you would like to see the effect of sending data to the 'input area' of the screen try:

```
10 OPEN $5,"K"
20 PRINT $5;RND
30 GOTO 20
```

You should see random numbers printed on the screen starting from the bottom and scrolling up. The program will end with an OUT OF SCREEN error message as the input area of the screen will not scroll in the same way as the normal print area.

Although in principle each stream has both an input and an output side, in practice the only channel that can accept both input and output is the keyboard channel. The other two, screen and printer are output only channels and any attempt to read data from them produces an error report J. Notice that this restriction is entirely a feature of the hardware that the stream is attached to.

You can associate more than one stream with any given channel but if you want to change the channel that a stream is associated with then its current association must be removed by CLOSEing it. The ZX BASIC command:

```
CLOSE $s
```

will remove any existing association between the stream 's' and a channel. In this sense CLOSEing is the reverse of OPENing a channel. CLOSEing a stream can also be used to inform the hardware component of a channel that it is no longer required by the stream and any 'cleaning up' operation that it needs should be carried out ready for another channel to use it.

mention of this flexible I/O que".

The default system

The four streams 0 to 3 are automatically OPENed by the Spectrum as part of its set up procedure. Initially the stream to channel assignments are as follows:

stream	channel
0	K
1	K
2	S
3	P

So even without an OPEN command:

```
PRINT $2;"HI THERE"
```

will print onto the screen. These streams are used by the Spectrum to direct program data to the correct device, for example, an LPRINT sends data to stream 3. These assignments of streams to channels can be changed using OPEN commands but the streams themselves cannot be CLOSEd. An attempt to CLOSE one of the default streams results in it being re-OPENed to its initial channel, as given in the previous table.

Memory formats

If streams and channels are going to be used to interface additional devices then it is obviously important to know the internal formats used to store channel information and which channel is associated with which stream.

The information that defines each channel is stored in the channel information area starting at CHANS and ending at PROG-2 (where CHANS and PROG are both system variables). Each channel has a separate 'channel record' which has the following format:

address	size	
n	2 bytes	address of output routine
n+2	2 bytes	address of input routine
n+	1 byte	channel code letter

where the input and output routines are machine code subroutines. The output routine must accept Spectrum character codes passed to it in the A register. The input routine must return data in the form of Spectrum character codes and signal that data is available by setting the carry flag. If no data is currently available then this is signalled by resetting both the carry and the zero flag. If the channel cannot support one of input or output then the address for the routine that performs the illegal operation should be set to an error handling routine. The standard way of handling errors in ZX BASIC is via a Restart call to address 8. In Z80 assembly language this amounts to:

```
ERROR  RST 0008
       DEFB errocode
```

where 'errocode' is the numeric code of the report to be given to the user.

The channel records for the standard three Spectrum channels and an additional channel that has not yet been described are as shown in **Table 1.**

TABLE 1 Details of the Spectrum's channel records.

address		keyboard channel record
CHANS		address of lower screen printout routine
+2		address of keyboard input routine
+4	K	channel K identifier
		screen channel record
+5		address of screen printout routine
+7		address of error routine
+9	S	channel S identifier
		editing buffer channel record
+10		address of buffer input routine
+12		address of error routine
+14	R	channel R identifier
		ZX Printer channel record
+15		address of ZX Printer routine
+17		address of error routine
+19	P	channel P identifier

Notice that each channel record is in the standard format as described and that the only channel that can support both input and output is the K channel. The new R channel is used internally by the Spectrum to send data to the editing buffer. The OPEN command will not allow the user to associate a stream with the R channel and so its use is limited.

Important Note — the format of a channel record is different when the Microdrives and Interface 1 are in use.

The association of streams with channels is stored in the system variables area of memory in the 38 byte area starting at STRMS (address 23568). That is, the stream table and each pair of bytes in this table holds a number 'x' that represents the address of the start of a channel record. Rather than simply storing the address of the channel record 'x' is the 'distance' that the channel record is away from the start of the channel information area —

address of start = address of
                        channel + x − 1
of channel record   area

That is, entry in the stream table is one more than the number of memory locations that the channel record is offset from the start of the channel information area. As there are a maximum of 16 streams you would think that a maximum of 32 bytes (ie one channel address per stream) would be sufficient to store all of the channel and stream associations. In fact the extra six bytes are used to store channel information for three internal streams corresponding to stream numbers 255, 254 and 253. These three internal streams are automatically associated with channels R, S and K respectively and as stream numbers are restricted to the range 0 to 15 they are inaccessible from ZX BASIC. However, the presence of these three internal streams does have to be taken into account when trying to find the address of the channel record corresponding to any of the streams 0 to 15. The first three entries in the stream table are for the internal streams 253 to 255, the fourth entry gives the address of the channel record to be used with stream 0 and so on. This means that the start of the stream table, as far as external streams are concerned, is

STRMS+6 or 23574

and the address of the start of the channel record associated with stream s (s in the range 0 to 15) is stored in the two memory locations starting at:

23574+s*2

"... system of streams and channels is a surprise bonus".

When a stream is OPENed to a particular channel the OPEN command stores the difference between the start of the channel record and the channel area itself plus one in the correct position in the stream table. When an INPUT or PRINT command sends data to a particular stream the stream table is examined to find the address of the channel record. When a stream is CLOSEd the number stored in its entry in the stream table is set to zero. Thus a zero entry in the stream table is used to detect an attempt to use a stream that hasn't been opened yet. Seven streams are automatically OPENed, the three internal streams and streams 0 to 3 as already described.

This system of channel records and the stream table is extended for use by the Microdrives but its essential features remain the same. Each channel is described by a channel record and streams are associated with channel records by use of the stream table.

Before moving on to consider using the knowledge of the stream and channel I/O it is worth mentioning the only other system variable that is connected with channel I/O — CURCHL. Each time a stream-oriented I/O command is used the stream number is used to look up the address of the channel record in the stream table. This address once found is then stored in the system variable CURCHL to direct all of the data produced by the I/O command to the correct channel. Thus following a command such as PRINT $s CURCHL contains the address of the start of the channel record associated with stream 's'.

Your own channels

The usual way of providing software to handle special I/O devices is to write BASIC subroutines using IN and OUT to send and receive data directly to and from the I/O ports that the device is allocated. For example, if a sound generator was allocated port 31 for its frequency control register then

OUT 31,f

would send the data (in the range 0 to 255) to the sound generator and so set its frequency. For simple devices and devices that are controlled by individual bits in the data the IN and OUT are very suitable. However, if the device is 'character-oriented', that is it receives and sends data in the form of characters, then IN and OUT are inadequate. For example, a parallel printer and a modem are both character-oriented devices and the best way to deal with them is via the usual PRINT and INPUT statements. Even if suitable subroutines could be written using OUT and IN to send and receive numeric and string data it is difficult to see how they could be used to LIST programs to the new devices. Clearly the way to go about providing a software interface to new character-oriented devices is via streams and channels.

Adding a character-oriented device to ZX BASIC's system of streams and channels can be done in one of two ways, either by changing the addresses stored in an existing channel record or by creating a completely new channel record.

The first method involves POKEing new addresses into an existing channel record that (point) to your own machine code routines positioned somewhere in memory. For example, suppose you want to interface a full sized printer in place of the ZX Printer, then changing the address stored in the first two locations of the ZX Printers channel record to point to your own printer driver output routine will make the commands LPRINT and LLIST, as well as any I/O commands referring to streams OPENed to channel P, send their data to the new printer. Writing the new printer driver is easy in principle, all it has to do is accept ASCII character codes in the A register and use these to print the correct ASCII characters on the printer. However, the Spectrum's character set includes many characters that the standard ASCII character set lacks and these would have to be detected and correctly interpreted by the driver. For example, all of the position and attribute control items within a PRINT statement will be sent to the printer driver as control codes as listed in Appendix A of the Spectrum manual. That is LPRINT TAB(10); sends ASCII codes 23, 10 and 0 to the printer driver. The 23 is the Spectrum's control code for TAB and the following two codes are the least and most significant byte of the parameter of the TAB function. It is important to realise that all of the Spectrum's output is converted to a stream of ASCII characters and codes before it is printed on the screen. This makes it possible for a printer driver to respond to or ignore all of the Spectrum's position and attribute control items as desired. For example, if the new printer was a colour printer then it could change the printing colour is response to —

LPRINT INK 3;"Hi there"

which sends the ASCII codes 16 (for INK) followed by 03 (for colour 03) to the printer driver.

As an example of this method of interfacing a new I/O device, the following program changes the output address stored in the P channel to refer to a machine code routine stored in the printer buffer area of memory. (Notice that this only works because the ZX Printer is not in use!). The new machine code output routine doesn't do anything really useful with the data that it receives in that it sends it to I/O port 254 which controls the speaker and border colour but this at least ensures that its effects can be seen and heard! The Z80 assembly language for this simple driver is shown in **Table 2.**

TABLE 2 Assembly language version of a simple driver.

address	assembler		code	comment
23296	outdrv	LD BC,254	01,00,254	load BC reg with 254
23299		OUT (c),A	237,121	send A register to port 254
23301		RET	201	return to ZX interpreter

This is used in the ZX BASIC listing shown in **Program 1.**

The machine code output routine is stored in the DATA statement in line 10 and loaded into memory by lines 20 to 50 (23296 is the start of the ZX Printer buffer). Subroutine 1000 changes the address in the channel record for channel P. Line

PROGRAM 1

```
  10 DATA 01,00,254,237,121,201
  20 FOR a=23297 TO 23301
  30 READ d
  40 POKE a,d
  50 NEXT a
 100 GOSUB 1000
 110 FOR i=0 TO 7
 120 LPRINT i;
 130 NEXT i
 140 GOTO 110
1000 LET c=PEEK 23631 +256*PEEK 23632
1010 LET c=c+15
1020 POKE c,23296−INT(23296/256)
1030 POKE C+1,INT(23296/256)
1040 RETURN
```

1000 gets the address of the start of the channel area into c and then line 1010 finds the start of the channel record for channel P. Lines 1020 and 1030 POKE the address of the new output routine into the first pair of locations in the channel record. If you enter and run this program you will see the border flash and change in a very wild manner. If you break into the program, you will obtain further proof that the new output routine is sending data to the border control port by LLISTing the program to it. (Note: disconnect the ZX printer before trying this program).

Changing the addresses stored in existing channel records is an easy method of adding new devices but it does have the disadvantage of removing one of the Spectrum's existing I/O devices. In practice it is impossible to modify channel K (the keyboard's channel record) because its I/O addresses are restored each time an INPUT statement is executed. This leaves the channel records for channel S and channel P as the only candidates for modification and as channel S is far too useful the only real candidate is P. This is fine as long as you don't want to use more than one extra I/O device and you don't want to use the ZX Printer at the same time.

To add any number of extra I/O devices it is necessary to add new channel records. Adding a new channel record sounds very easy but there are a few minor details that have to be taken into account. Firstly, it is possible to create a channel record anywhere in memory, not just in the channel information area, but if the channel record is stored above the INPUT workspace area (starting at WORKSP) the CURCHL (current channel) system variable will be altered as memory is added to the workspace area during an INPUT command. This of course will mean that the current of location of the channel record will be lost and the Spectrum will crash. However, if the channel record is stored below the INPUT workspace area then everything works correctly. In the demonstration given below the ZX printer buffer is used to store both the new channel record and the new I/O routines. A second difficulty is that the OPEN and CLOSE commands will only work with the standard channel records for K,S and P. This means that as well as providing a new channel record and I/O routines you also have to provide a subroutine to open the channel to any stream and if necessary a subroutine to close it. Putting all this together gives the Z80 assembler for the channel record and I/O routines shown in **Table 3.**

TABLE 3 Channel record and I/O routines.

address	assembler		code	comment
23296	chanrec	DEFB 0	0	l.s.b. of output address
23297		DEFB 91	91	m.s.b. of output address
23298		DEFB 11	11	l.s.b. of input address
23299		DEFB 91	91	m.s.b. of output address
23300		DEFB "E"	69	channel identifier
23301	outdrv	LD BC,254	01,00,254	load BC reg with 254
23304		OUT (C),A	237,121	send contents of A to 254
23306		RET	201	return to ROM code
23307	indrv	RST 8	207	error restart
23308		DEFB	18	invalid device error code

The first five bytes form the new channel record. The routine starting at 23301 is the output routine and this simply sends the code in the A register to output port 254, which is the speaker and border colour port. The routine starting at 23307 is the input routine and this simply reports an error to indicate that input is not allowed with this channel. Obviously in a real application either routine could be very much more complicated. **Program 2.** BASIC program using this machine code.

PROGRAM 2.

```
  10 DATA 0,91,11,91,69,1,0,254,237,121,201,207,18
  20 FOR a=23296 TO 23308
  30 READ d
  40 POKE a,d
  50 NEXT a
 100 LET s=5:GOSUB 1000
 110 PRINT $5;RND;
 120 GOTO 110
1000 LET a=23574+2*s
1010 LET c=PEEK 23631+256*PEEK 23632
1020 LET r=23296−c+1
1030 POKE a,r−INT(r/256)*256
1040 POKE a+1,INT(r/256)
1050 RETURN
```

Lines 10 to 50 load the new channel record and I/O routines into the ZX printer buffer. Subroutine 1000 will open stream s to the new channel. In other words it is the equivalent of OPEN $s,"E". Line 1000 works out the correct address for stream s in the stream table. Lines 1010 and 1020 work out new channel records offset from the start of the channel information area (plus one) and lines 1030 and 1040 store it in the stream table. Line 100 uses subroutine 1000 to open stream 5 to device E and lines 110 to 120 provide a demonstration by sending the codes corresponding to random numbers to the sound and border control port. A further demonstration can be gained by stopping the program and typing LIST $5 which produces a flash of colour and sound indicating that the program has been listed to port 254! If you change line 110 to read

```
110 INPUT $5;i
```

then you will get the correct error message indicating that this channel cannot be used for input.

Apart from having to write more comprehensive and specialised I/O drivers there is nothing difficult about adding new channel records to ZX BASIC. Notice, however, that the above program will not work if the Microdrives are connected.

The problems of writing an output driver have already been described but before bringing this article to a close it is worth mentioning the extra requirements for an input driver. If an input channel is going to supply a single character code, as an eight-bit A to D converter might, then the best BASIC command to use is INKEY$ # which will return a single character. However, if you are planning to use INPUT # to read in a collection of character codes then you have to be aware of two things. Firstly, INPUT statements also perform output by printing prompts etc. and it is not enough to make the output routine an error return, you have to handle any data that the INPUT statement sends even if you simply ignore it! Secondly, an INPUT # statement accepts data as if it was being typed at the keyboard. This means that if you use INPUT #s;i to read in a number to the variable i then the device driver has to send a collection of ASCII codes corresponding to digits and ending with an ENTER code, just as a number would be entered from the keyboard. Finally, notice that even when reading data from a special piece of hardware the INPUT command will interpret editing codes, delete etc., correctly! The best way to think about it is that INPUT always works as if it was receiving a stream of character codes corresponding to keys that are being pressed on the keyboard.

Conclusion

The Spectrum's system of streams and channels is something of a surprise bonus to the ZX BASIC programmer. Used within programs it provides the advantage of device independence and an overall increase in flexibility with no disadvantages. To the Z80 assembly language programmer streams and channels provide the ideal way of providing software interfaces with any new equipment. The examples given in this article provide a starting point for any real software interfaces that you might want to write. ■

Spectrum BEEP command

Derek Harding shows how to overcome the problem of implementing the Spectrum BEEP command from machine code.

The BASIC "BEEP" command on the Spectrum is not easy to implement from machine code because the duration of each beep depends on its frequency (tone) and so this short routine was evolved; a much modified version of a routine I found years ago. Once the duration of a note is set, it will remain that length regardless of the frequency until such time as it is changed by the programme. From BASIC it minimises that annoying break in sound as fresh data is retrieved to give the "BEEP" parameters.

Listing 1 is the assembler listing for this routine, which can be located anywhere convenient in the memory since it uses no addressed jumps, or calls. (ie no GOTOs or GOSUBs). **Listing 2** repeats the listing, and shows in the left hand column the address (in hex) to which I assembled it. The second column gives the hex code for the mnemonic in the third and fourth columns. At the bottom of Listing 2 is a hex dump of the routine, for those who do not have an assembler.

Listing 3 repeats the routine, except that after the 'pops' it goes into another routine. This programme will keep running quickly up the scale until the SPACE key is pressed. It could, therefore, be used at the end of a game, etc. Again, it was written to the address 8000H, but can be re-located to suit. The start of the programme in **Listings 1** and **2** is to the address of "PUSH AF" (8000H in the listing), and for **Listing 3** the start is to "LD HL,1870" (8021 in the listing). The short BASIC programme in **Listing 4** is intended to give some feel of how the routine (of Listings 1 and 2) can be used. Make sure that you have loaded the routine first, and altered the USR address to suit your memory first!

Now to the programme itself. Firstly, it may be that the CPU registers are holding information which is to be retained, so each of the registers is pushed onto the stack. When that is complete, the interrupts are disabled to stop the note which is to be played from suffering from a buzz (as the keyboard is scanned). For this reason, no keys are active whilst a note is being played. The A register is loaded with zero in preparation for the rest of the routine. The length of the note to be played is entered into the C register. The following six lines set up a loop which sets the duration of the note, and checks to see if the end has been reached. The D register holds the frequency of the note to be played (its pitch). The note is sounded by switching the loudspeaker (output OFEH) on and off each alternate cycle. The action of the XOR 10 command is to set the port on or off, depending upon what it was before, while leaving the rest of the A register unaffected; so if a value other than zero was set up at the beginning, it will only change the fourth bit. Having turned the speaker on or off, the loop is continued until the appropriate duration is complete. The action then moves to the last few lines where the interrupts are re-enabled (so allowing the keyboard to be active) and the registers are popped off the stack to return them to their original values before RETurning to the main programme.

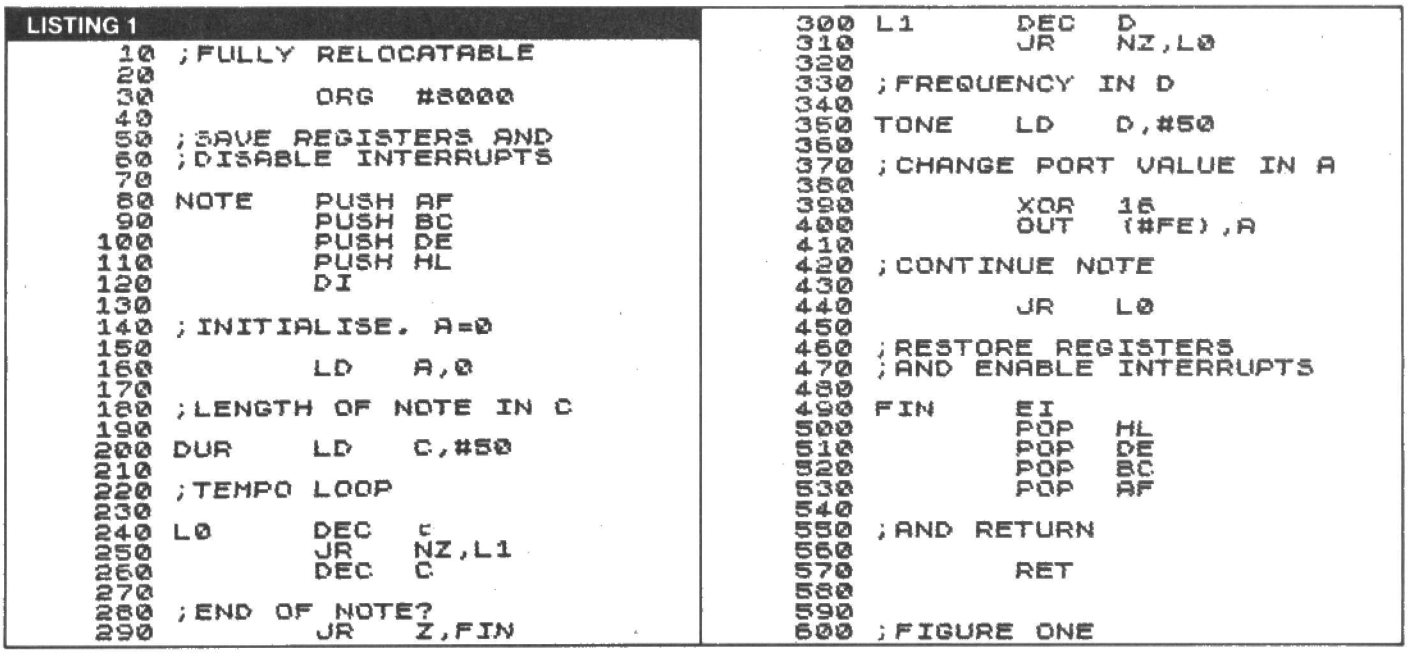
Referring again to **Listing 4,** in line 10 a series of data statements appear. The first item is the number of items following, and is used to set up the READ loop. However, the following numbers represent a span of four octaves, and approximate to the key of E flat major. If other keys are required, then sharps are found by interpolating, and the number of complete octaves is obviously reduced.

To recap, the C register holds the duration of the note, and the D register holds the pitch or frequency. The values required are poked into the 9th and 20th byte of the routine respectively.

If you do not understand machine code at all, then choose where you wish to load the code to (eg USR "a" could be appropriate), and call that address START. From there, poke the decimal representation of each number in the hex dump into memory sequentially from START. Poke START + 8 with the duration, poke START + 19 with the pitch, and to use the routine then RANDOMISE USR START. If you are using the second routine, that shown in **Listing 3,** the RANDOMISE USR START + 33 line should be used.

Finally, the value subtracted at 802B in **Listing 3** (04 in the listing) determines how quickly the notes go up the scale. Other values can be poked in, but to make sense they should be divisors of 256. ■

LISTING 1



```
 10 ;FULLY RELOCATABLE
 20
 30          ORG   #8000
 40
 50 ;SAVE REGISTERS AND
 60 ;DISABLE INTERRUPTS
 70
 80 NOTE     PUSH  AF
 90          PUSH  BC
100          PUSH  DE
110          PUSH  HL
120          DI
130
140 ;INITIALISE. A=0
150
160          LD    A,0
170
180 ;LENGTH OF NOTE IN C
190
200 DUR      LD    C,#50
210
220 ;TEMPO LOOP
230
240 L0       DEC   c
250          JR    NZ,L1
260          DEC   C
270
280 ;END OF NOTE?
290          JR    Z,FIN
300 L1       DEC   D
310          JR    NZ,L0
320
330 ;FREQUENCY IN D
340
350 TONE     LD    D,#50
360
370 ;CHANGE PORT VALUE IN A
380
390          XOR   16
400          OUT   (#FE),A
410
420 ;CONTINUE NOTE
430
440          JR    L0
450
460 ;RESTORE REGISTERS
470 ;AND ENABLE INTERRUPTS
480
490 FIN      EI
500          POP   HL
510          POP   DE
520          POP   BC
530          POP   AF
540
550 ;AND RETURN
560
570          RET
580
590
600 ;FIGURE ONE
```

LISTING 2

```
8000    F5              PUSH  AF
8001    C5              PUSH  BC
8002    D5              PUSH  DE
8003    E5              PUSH  HL
8004    F3              DI
8005    3E00            LD    A,#00
8007    0F50            LD    C,#50
8009    05              DEC   B
800A    2003            JR    NZ,#800F
800C    0D              DEC   C
800D    280B            JR    Z,#801A
800F    15              DEC   D
8010    20F7            JR    NZ,#8009
8012    1650            LD    D,#50
8014    EE10            XOR   #10
8016    D3FE            OUT   (#FE),A
8018    18EF            JR    #8009
801A    FB              EI
801B    E1              POP   HL
801C    D1              POP   DE
801D    C1              POP   BC
801E    F1              POP   AF
801F    C9              RET
8000 F3  F5  C5  D5  E5  3E  00  0E
8008 19  05  20  03  0D  28  0B  15
8010 20  F7  16  14  EE  10  D3  FE
8018 18  EF  E1  D1  C1  F1  FB  C9
```

LISTING 3

```
8000    F5              PUSH  AF
8001    C5              PUSH  BC
8002    D5              PUSH  DE
8003    E5              PUSH  HL
8004    F3              DI
8005    3E00            LD    A,#00
8007    0E18            LD    C,#18
8009    05              DEC   B
800A    2003            JR    NZ,#800F
800C    0D              DEC   C
800D    280B            JR    Z,#801A
800F    15              DEC   D
8010    20F7            JR    NZ,#8009
8012    1600            LD    D,#00
8014    EE10            XOR   #10
8016    D3FE            OUT   (#FE),A
801D    C1              POP   BC
801E    F1              POP   AF
801F    1807            JR    #8028
8021    217018          LD    HL,#1870
8024    7C              LD    A,H
8025    320880          LD    (#8008),A
8028    F3              DI
8029    7D              LD    A,L
802A    D604            SUB   #04
802C    321380          LD    (#8013),A
802F    6F              LD    L,A
8030    FE00            CP    #00
8032    20CC            JR    NZ,#8000
8034    FB              EI
8035    3E7F            LD    A,#7F
8037    DBFE            IN    A,(#FE)
8039    1F              RRA
803A    38E5            JR    C,#8021
803C    C9              RET
8000 F5  C5  D5  E5  F3  3E  00  0E
8008 18  05  20  03  0D  28  0B  15
8010 20  F7  16  00  EE  10  D3  FE
8018 18  EF  FB  E1  D1  C1  F1  18
8020 07  21  70  18  7C  32  08  80
8028 F3  7D  D6  04  32  13  80  6F
8030 FE  00  20  CC  FB  3E  7F  DB
8038 FE  1F  38  E5  C9  00  00  00
```

LISTING 4

```
   5 POKE 32776,0: POKE 32787,1:
 RANDOMIZE USR 32768: FOR g=128
TO 16 STEP -16: POKE 32776,g: RE
STORE : READ l: FOR f=1 TO l: RE
AD a: POKE 32787,a: RANDOMIZE US
R 32768: NEXT f: NEXT g
  10 DATA 29,248,221,197,185,165
,147,131,123,110,97,92,82,72,64,
61,54,48,45,40,35,31,29,26,23,23
,19,17,15,14
```

E&CM PCB SERVICE

April 1983	
TV to RGB Conversion	£2.70
July 1983	
Power Control for Micros	
Relay Board	£2.02
DAC Board	£1.77
Stepper Motor Driver	£1.59
BBC Sequencer Interface	£2.10
August 1983	
Oric Output Port	£2.10
Spectrum Sound Board	£2.20
September 1983	
BBC Darkroom Timer	£1.15
Cassette Signal Conditioner	£1.23
ZX81 Sound Board	£3.77
October 1983	
Spectrum Effects Box	£1.71
Cassette Signal Conditioner	£1.23
BBC EPROM Programmer	£5.12
November 1983	
Lie Detector Interface	£1.88
Microcontroller	£2.13
ZX Light Controller	£4.28
December 1983	
BBC Sideways RAM	£4.98
Electron A/D	£2.91
January 1984	
Electron I/O Port	£2.32
February 1984	
BBC Speech Synthesiser	£4.53
Electron RS432	£2.70
Spectrum Speech Board	£3.22
BBC Sideways ROM Board	£5.48

HOW TO ORDER

List the boards required and add 45p post and packing charge to the total cost of the boards. Send your order with a cheque or postal order to:

E&CM PCB Service, 45 Yeading Avenue,
Rayners Lane, Harrow, Middlesex HA2 9RL
Telephone: 01-868 4854

Please supply the following PCBs:

....

....

Post & Packing 45p

TOTAL £ ________

Signed .... Date ....

Name .... (please print)

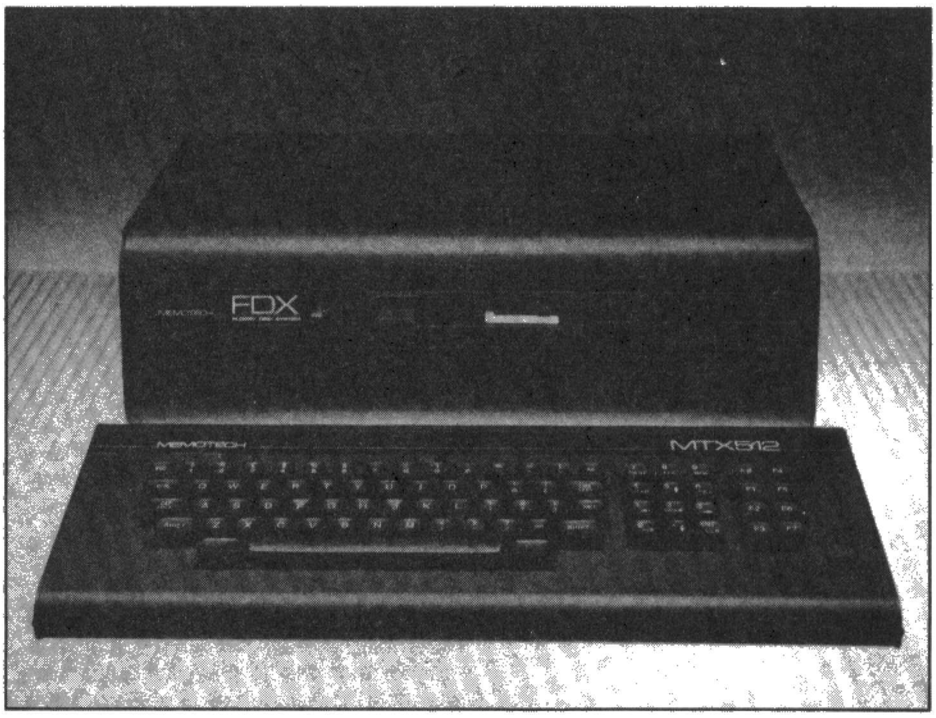
Address ....

....

....

PLEASE ALLOW 28 DAYS FOR DELIVERY

MTX 500... rivals the BBC



Do not be deceived by the quiet and unpretentious launch of the MTX 500. The specifications are so good, says Richard Sargent, that the competition may as well resign and go home.

The Memotech 500 has planted itself firmly in the market between the Electron and Commodore 64 on the one hand, and the BBC micro on the other, and offers a range of facilities available on none of these machines.

Those familiar with Memotech's ZX81 peripherals will know that the company has an eye for good design and does not skimp on the materials it uses. The MTX is in a case with a long, narrow design which allows plenty of space for an extended keyboard on its top surface and a plethora of ports along its rear edge. The case is made from extruded aluminium and to call it sturdy would be something of an understatement. Fortunately it isn't too heavy, since the transformer is housed in a separate (plastic) case. Memotech are hoping that vast numbers of these computers will find their way into schools and colleges, and their rugged construction will be of great benefit in those conditions.

The Keyboard

The keyboard is extremely high quality: there is a good feel to all the keys, and no gaps around them into which paper clips and the like might fall. The metal case helps to suppress radio-frequency emissions, should this concern you, but it also acts as an enormous heat sink thus allowing the components to operate at an even temperature without recourse to the use of a fan. The keyboard has a separate cluster of cursor control and other editing keys, which can double as a numeric keypad, and yet another group of keys give 16 user-defined functions. All keys have auto-repeat and there is a CAPS LOCK key as well. The RETURN key is only a little larger than the normal keys. Most of the sockets on the MTX lie in a plastic housing which runs the length of the back of the machine. Deeply recessed into the plastic — too deeply in some cases — are sockets for two Atari-type joysticks, a Centronics standard parallel printer, cassette "ear" and "mic", UHF output, 6 pin DIN for power, audio output and composite video output. Two blanks exist for the twin RS232 ports which are optional extras. Communication to the MTX disc system and other extras is by means of a parallel ribbon cable and the connector for that is on the RS232 board. There is spare capacity inside the case for up to two boards which can be plugged into an edge connector on the main PCB; they can be RS232 and/or dynamic RAM boards. There is an empty IC socket on the main PCB and this is in fact another port, consisting of 8 input and 8 output lines complete with their strobe signals. On the left hand side of the case, behind a removable cover, is another edge connector. This is intended for add-on ROM packs which would hold other high level languages. Like the BBC's sideways ROM, they can be called up by direct commands.

The entire MTX case is hinged along the front edge, and it opens into two halves. To open it up six screws holding two metal end plates have to be removed, but that is all: nothing needs to be unplugged. The PCB is fully populated with ICs, over 40 of them in fact, and there are no flying leads or patches in evidence. It is a very neat board. The components are standard TTL and memory chips — there are no ULAs — and only the key ICs are socketed, namely the Z80A, the Z80 CTC timer, the Texas Instruments video processor, a small memory-paging ROM and the three 8K ROMs.

Software

There proved no problems in getting the MTX running, despite the fact that the manual was very sketchy on the entire subject of tape interface. To be fair, the manual I had was a pre-production version and it may be that I didn't have the section relating to SAVEing LOADing and files. The baud rate of the cassette port is supposed to be software programmable, but I couldn't find the commands you would use to change it. However, the default rate is 2400, and the program I chose to load fairly zipped into the machine (12K in 60 seconds) and then auto-ran. It was a Frogger-type game, one of two supplied by Continental Software as a free introduction to MTX software. The other game was Draughts. The colour on the MTX is very good indeed, though the set (a Hitachi portable TV) did need re-tuning after about half an hour of use. Afterwards the picture remained stable. In the case of Draughts it was noticeable that the leading character of a line was well over in the lefthand edge of the television screen and couldn't be read. Swopping to an older TV which had horizontal hold brought this character inwards, and both ends of every line were well on the screen. Memotech have tried to help with this problem of lack of horizontal adjustment by writing most of their software with a space in the first position of the line, so listings, error reports and so on are always readable.

Sound

The MTX has three sound channels and one noise channel, and the sounds may be heard either through the TV speaker or through an external amplifier. The SOUND command can be used in two ways. In direct mode a single note is played until a subsequent command turns it off. In continuous mode sequences of notes can be played by loading them into a buffer. Naturally the sound buffer requires memory so the SDBUF command is used to put aside sufficient space in the same way that DIM works for ARRAYS. The commands are sufficiently complex to enable the computer to be used as a synthesiser, and it would have been nice to have had a good demonstration of its capabilities on tape. There is a demonstration tape supplied with the machine and I can only say that I hope it isn't the final version since it doesn't begin to do justice to the power of the computer.

"a built in Z80 assembler/disassembler and a powerful machine code monitor"

Graphics

The resolution of the MTX graphics screen is 256 x 192 pixels, which is the same as the Spectrum and slightly less than the Commodore 64 and the RML 380Z. Inevitably comparisons will be drawn between the screen capabilities of the MTX, the Commodore 64 and the BBC micros. All these machines offer 16 colours. The 64 has a dot resolution of 320 x 200 and the BBC a dot resolution of 640 x 256. However, this is at best a crude comparison. The BBC high resolution mode 0 only gives 2 colours. Their mode 2 is the one most like the MTX display, and it takes 20K of memory away from the user, whereas the MTX graphics take no memory away from the user. The MTX text mode is 40 x 24 characters, and there is no teletext mode. The facilities offered by the three graphic modes are shown in **Figure 4.**

Both the 64 and the MTX have sprites. A sprite is a programmable object made up of pixels that can be made into just about any shape. The object can be moved easily around the screen by commands, and once moving will continue to do so independently of the program until new commands are issued. Sprites can change colour, can flag a collision condition with another sprite, can be made to pass in front and behind others and can be expanded in size. Whereas the 64 has 8 sprites which you tediously control with POKEs, the MTX has 32, controlled easily by special BASIC commands.

MTX BASIC

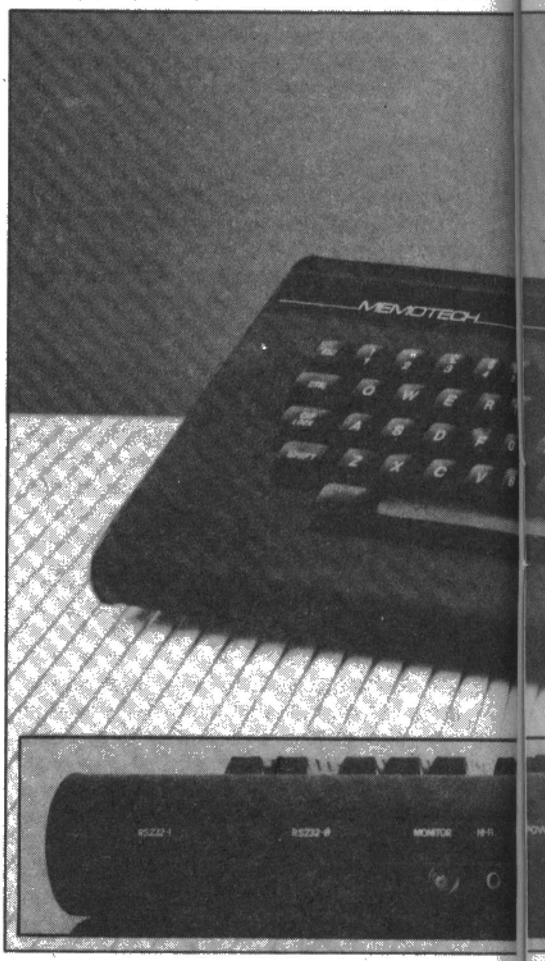
A closer look at the MTX BASIC (see **Figure 2**) shows that it is more or less a Microsoft Basic to which a whole range of specialised commands have been added, most of these being powerful graphic handling commands. GENPAT, for example may be used to redefine the entire ASCII character set, or to create 26 user definable shapes, or define a sprite shape.

ATTR determines the effects on the graphics screen of using one of the plotting commands such as PLOT, DRAW or ARC, and provides facilities such as inverse printing and over-printing. PHI is another interesting command which changes the direction of a drawn line by a relative angle, and in association with a few other commands such as DRAW and ANGLE it enables a BASIC program to emulate some of the commands of Turtle Logo. This may or may not help the MTX to gain a foothold in the classroom, but Memotech's innovation in software doesn't stop there. They have invented a new language called NODDY **(Figure 3)** which will undoubtedly assist in the writing of textual programs and interactive programs. It can be used from within BASIC as an easy way of creating pages of text or as a language in its own right.

Machine code

The MTX has a built in Z80 assembler/disassembler and a powerful machine code monitor that should please a great many advanced programmers. Even the provisional manual, which runs to some 250 A4-size pages, has a wealth of detail for the machine-code specialist with time on his hands. Details of port decoding, interrupt vectors, annd the TMS9929 (video) and SN76489A (sound) CHIPS are given, though information on the Z80CTC is missing, which is a pity since it is used (amongst other things) to drive a real-time clock which can be read by BASIC in hours, minutes and seconds format.

The assembler is a small but sturdy piece of software which is designed to be used from within BASIC, so it is intended that it should be used to write short sections of code rather than complete programs. However, Memotech has devised a novel way of saving memory space with the assembler, so longer programs *could* be written if so wished. If machine code is required at BASIC line 100 direct command ASS.100 is entered to pass into a true mini-assembler. The source code is not entered after a BASIC line number, and is in fact not stored in memory at all. When leaving assembler (which is done by pressing CLS,RETURN) the object code and a few other labels and control information exist at line 100 correctly assembled and ready to run. Source code may be inspected, and even dumped to a printer, but what is seen is a disassembly of the stored object code. The advantage in terms of memory space with this system is tremendous but there is one drawback. If any lines beneath 100 are edited, the position of the machine code will shift and many vital addresses will be rendered invalid: you would have to re-enter the source code and reassemble it! The golden rule would be — keep all code at the beginning of a BASIC program. If this doesn't quite appeal to you, remember that the more usual USR command exists, and it can be contrived to place code in high memory using some other method than the built in assembler.



The Memotech MTX 500 has 12 numeric and 8 func

CPU	4MHz Z80A
Memory	32K user RAM, with optional extension 24K ROM, with optional extension 16K Video RAM
Keyboard	79 keys including function keys. Auto repeat and caps lock
Screen	TV or Monitor (composite video 40 x 24 or 32 x 24 text 16 colours, 256 x 192 pixels
Cassette	2400 baud software selectable
Ports	Centronics, 2 joysticks, 16 I/O RS232 is optional
Disks	Optional 5.25", 8" or Winchester, with CP/M 2.2, 80 columns & teletext
Languages	BASIC, NODDY, ASSEMBLER Optional high-level languages in ROM

Figure 1. Technical data

tion keys. Inset, the back of the case showing output and extension ports.

ABS	ASC	ATN	LN	CHR$	CLEAR
CLS	CONT	COS	DATA	DIM	EDIT
EXP	AND	GOSUB	GOTO	INKEY$	IF
THEN	ELSE	LEFT$	LEN	LET	LIST
LLIST	LOAD	LOG	MID$	NEW	FOR
STEP	NEXT	PEEK	POKE	PRINT	GOSUB
RAND	READ	REM	RESTORE	RETURN	GOTO
RIGHT$	RND	RUN	SAVE	SGN	SIN
SQR	STOP	STR$	PI	TAN	USR
VAL	VERIFY	OR	NOT	MOD	LPRINT
BAUD	CLOCK	INK	PAPER	PAUSE	NODDY
PLOD	CSR	PANEL	SOUND	ASSEM	ATTR
COLOUR	EDITOR	DSI	SDBUF	TIME$	GR$
SPK$	ROM	CRVS	VS	VIEW	GENPAT
CIRCLE	DRAW	PLOT	ARC	LINE	ANGLE
PHI	SPRITE	MVSPR	ADJSPR	CTLSPR	AUTO
INPUT	INT	OUT	PAUSE	ON	

Figure 2. MTX BASIC — Commands and Functions

Graphic Mode 1

```
   ?       A colour pair
16666666   Two ink colours are
61666666   allowed for the 8 x 8
66166666   pattern: here a black
66616666   diagonal with a red
66661666   surround is shown.
66666166   The 256 patterns may
66666616   have a total of 32
66666661   different colour pairs.
```

Graphic mode — multicolour

```
6666  Any of the 16 colours may be used
6666  in this small square, and the
6666  occurrence of any colour anywhere
6666  on screen is unrestricted.
```

Graphic Mode 2 colour.

16 colours may be placed on any pattern position. The black diagonal line 11111111 crosses a multicoloured surround made up by the colours 9ABCDEF0. The backdrop colour is allowed to show through on the last line. The pattern library is now 768, rather than 256.

19999999	0	Transparent	1	Black
A1AAAAAA	2	Medium green	3	Light green
BB1BBBBB	4	Dark blue	5	Light blue
CCC1CCCC	6	Dark red	7	Cyan
DDDD1DDD	8	Medium red	9	Light red
EEEEE1EE	A	Dark yellow	B	Light yellow
FFFFFF1F	C	Dark green	D	Magenta
00000001	E	Grey	F	White

Figure 4. Colours available in the three graphics modes: one, two, and multicolour.

B	BRANCH	to a label.
I	IF	input = match then goto label.
G	GOTO	page at label if specified
E	ENTER	input into VS 7.
A	ADVANCE	to next program page on stack.
R	RETURN	to BASIC.
S	STACK	up program pages.
P	PAUSE	before continuing with program.
L	LIST	a Noddy page to printer.
O	OFF	Remove a program page off the stack.
D	DISPLAY	a Noddy page on VS 5.

Figure 3. NODDY's eleven commands

Expansion and potential

The MTX expansion potential is well thought out. The key to both the MTX RING System and to the disc drive Systems is the Communications (RS232) board mentioned earlier. It costs £60 inclusive of VAT, which is just £10 more than Sinclair's ZX Interface 1, and yet it has *two* RS232 channels, both with full handshaking facilities and modem communication lines. In order to upgrade to discs, you need an MTX with 64K of user RAM, already available on the MTX 512 (£315) or on an upgraded MTX 500 (£275+£50). The cheaper Memotech disc system is the FDX system with twin 5¼" Qume Drives (2 x 500K) and all the necessary bits and pieces to run CP/M with an 80 column colour screen. This costs £870 inclusive of VAT, which is keenly competitive with the Acorn and Research Machines disc systems which presently dominate the Educational market. When you consider the physical construction of the MTX and the software touches such as syntax checking at the time of input, the power to control sprites from BASIC and the bonus of the text-handling language NODDY, then you begin to realise that this is a machine that would have no difficulty fitting into the classroom if sufficient software were available "off the shelf" to support it.

As a business system at £1245 all told it has many rivals, but the pleasant keyboard and optional extras such as the 256K Silicon-disc may generate interest. As a hobby computer it has good sound and good graphics, and it is the sort of machine where the operating system is not going to remain a mystery, with all the work happening deep inside a ULA. It's a machine which will get things added to it, not because of any deficiencies in its own specification, but because it will be fun to let it drive robots or control the new video-disc players when they become available. ■

Move over BASIC

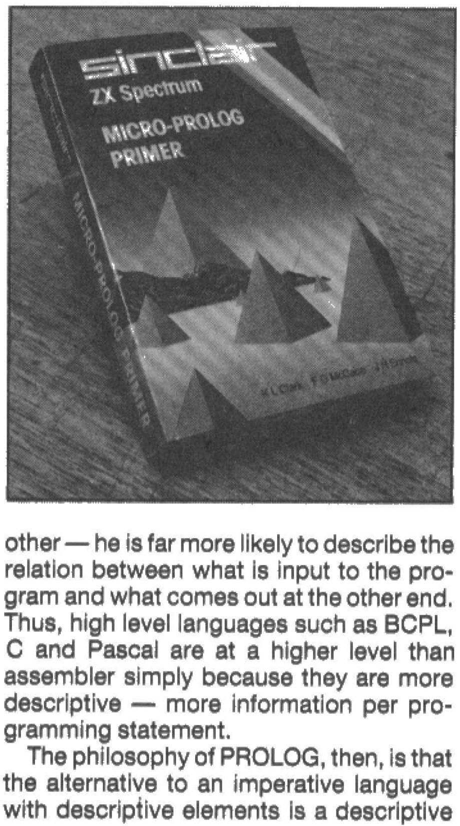
With the introduction of micro-PROLOG, a 5th generation language is now available on the Spectrum. Adam Denning explains the principles behind logical programming and assesses this new implementation.

Micro-PROLOG is described as a fifth generation language, and as the Spectrum is the first real home computer to have it, this must increase Sinclair's credibility no end. For an outlay of a mere £24.95, one can buy the cassette based version of the language, which is accompanied by a short booklet written by Jonathon Briggs, and a much larger manual called 'The micro-PROLOG primer', written by the implementers of the language.

These implementers are a company called Logic Programming Associates, based at Imperial College in London, and they more than almost anyone else are convinced of the virtues of micro-PROLOG. PROLOG stands for programming in logic, and was originally designed in 1972 by Colmeraur and Rousell in Marseilles. This was written in a now-outdated language, Algol-W, until rewritten the following year in FORTRAN, which is when it began to take off worldwide.

Very loosely based on LISP, PROLOG was different from all other languages in that rather than being an imperative language, like BASIC or assembler, it is a descriptive language. Expressed like that, it is a fairly hard concept to understand, so we will have to precis the publicity handout for the language. In all the common languages used today, the programmer must describe fairly precisely how a result is to be arrived at, rather than what it is that must be computed. A program written in one of these conventional languages consists of sections of instructions each of which describes an action to be performed by the computer. These languages are geared towards the description of the behaviour needed to achieve the result.

It is true that we occasionally think behaviourally, but in general, we don't — we are far more concerned with the 'What' than the 'How'.



If a human had to describe a program to a fellow being, it is very unlikely that he would list the program and pass it to the other — he is far more likely to describe the relation between what is input to the program and what comes out at the other end. Thus, high level languages such as BCPL, C and Pascal are at a higher level than assembler simply because they are more descriptive — more information per programming statement.

The philosophy of PROLOG, then, is that the alternative to an imperative language with descriptive elements is a descriptive language with imperative elements; a language in which programs are primarily descriptive definitions of a set of relations or functions to be computed.

"... a language in which programs are primarily descriptive functions ..."

The execution of a descriptive program is then a use of the definitions to find an output corresponding to a given input. The way in which the definitions are used in order to compute the output value gives each definition an alternative or control reading. By taking into account this control reading we might prefer one set of definitions to another, and we might improve the efficiency of the evaluation by adding extra control conditions to the definition which are ignored in the descriptive reading. These are the fundamentals of programming in a descriptive language. However, it is still the case that the program is primarily a description of what it is supposed to compute, rather than a prescription of how it should compute it.

With this view in mind, PROLOG was developed and quickly became available for a number of machines. Then, of course, the micro boom happened, and educationalists everywhere wanted this language on their machines. So the syntax of the formal language was changed somewhat, and micro-PROLOG was born. We are told that this syntax is the only difference between PROLOG and micro-PROLOG. Logic Programming Associates have also provided Acorn with a version of the language for the BBC machine, and this is currently undergoing evaluation.

Micro-PROLOG is becoming popular amongst hobbyists because of its purported suitability for expert systems. These expert systems are programs that are based around a generally huge database of facts and relations, such that upon being posed a question concerning these facts, an intelligent answer can be arrived at. It is a branch of artificial intelligence, and is used for such things as medical diagnostics, car mechanics, sophisticated accounting packages, and even the production of PROLOG compilers themselves.

It is perhaps this aspect of artificial intelligence that attracts the programmer more than anything else, while it is the notion of being taught to think logically that attracts the educationalists. One has to learn to stop thinking procedurally, ie how something can be done, and start thinking descriptively, that is, what is to be done?

Currently the most famous (and vastly over-rated) example of artificial intelligence is Weisanbaum's Eliza program. This was originally written in LISP, and to the ignorant user seemed like a reliable pyscho-analyst, giving intelligent and sensitive answers to the problems posed by the user. Of course, this was just a gigantic database of connected words and phrases, and simply by identifying various verbs and nouns within the input sentence, the program was able to substitute various set phrases. At the time, this seemed revolutionary, but of course this thing is now considered such a simple programming operation that there have even been two page BASIC versions of it published.

Since then, artificial intelligence has come a long way, to the extent where even the AA are considering using computers to help their mechanics' diagnostics on the road. PROLOG is destined to be the language of this principle, and micro-PROLOG the version that most of us will see.

The language is loaded into the Spectrum simply by typing LOAD"", (there is a BASIC interface program) and after about two minutes the up-until-now green screen develops a yellow border and micro-PROLOG introduces itself. With the interpreter loaded, one has some 25K spare, which although by no means enough to write an AA-style expert system, is easily sufficient for almost any programming application likely to be attempted on a Spectrum. For all the exercises in the introductory manual and most of the Primer, the user then has to load a new 'front end' called SIMPLE. This is done by typing LOAD SIMPLE (no horrible single keyword entry here!) and pressing ENTER. SIMPLE is a specially designed beginner interface for the language, and the rather formal syntax of micro-PROLOG is bypassed by it. Loading this section is a pleasure in itself, as Logic Programming Associates have taken lessons from Acorn when it comes to cassette interfaces. SIMPLE is loaded in short blocks (they appear to be 256 bytes at a time), with error checking on each block. If a block was loaded incorrectly, the computer tells you, and simply rewinding the tape to just before that block gives you the chance of trying again, without having to go through the rigmarole of starting right from the beginning. Excellent.

"...sufficient for almost any application likely on the Spectrum..."

Having loaded SIMPLE, it is then recommended that you work through Jonathan Briggs' introductory booklet, in which you learn such basic concepts as adding to the database and querying it afterwards. The instruction to add things to the database is, logically enough, 'add', and the syntax is as follows:

```
add(Adam writes-for ECM)
```

This tells the computer that argument one, Adam, is related to argument three, ECM, by the second argument — the **relation** writes-for. If we then add a further fact:

```
add(ECM pays Adam)
```

The arguments Adam and ECM are now related in two ways. If we wanted to see if Adam writes-for ECM, we have to ask it:

```
is(Adam writes-for ECM)
```

and the computer will reply (in this case) with 'YES'. However, if we didn't know exactly who (or what) Adam was related to by writes-for, we ask in a slightly different way:

```
is(Adam writes-for x)
```

This is simply to determine whether or not this relation is defined for Adam, as x is a variable. If the computer finds that somewhere within its database there is a fact that relates Adam to something else (it doesn't matter what) by writes-for, then it will reply 'YES', otherwise it will say 'NO'. To find out exactly what Adam is related to by writes-for, we again use variables, like this:

```
which(x : Adam writes-for X)
```

This will reply with all the facts in the database in which Adam is related to something by writes-for. In our rather simple example, this means:

```
ECM
No (more) answers
```

The line saying no more answers tells the user that the list above is all that it could find. In other words, it marks the end of the evaluation. Of course, nearly all databases will be much larger than this, and we may forget exactly what facts we have put in. Typing:

```
list all
```

will do just that, and the screen will fill up with all the facts previously entered. If we just wanted to know which relations have been defined, typing:

```
list dict
```

will tell us this.

So far, this is all well and good, but a language would need rather more power than this to be exceptional. In comes the concept of rules. If we add a further definition, this time called a rule, that says the pays relation is true if the writes-for relation is true, we have this:

```
add(x pays y if y writes-for x)
```

It is now obvious that ECM only pays Adam if Adam writes-for ECM. Unfortunately, this is a fact of life!

One rather interesting result of this fact-and-rule creation of database is that as we can query any argument, we can consider most relations as being reversible. Thus, if we were unintelligent enough to use micro-PROLOG to write a calculations application in any relational operator that has a complement, such as multiply, which is the opposite of divide, it only needs to be defined one way. Therefore, if we have multiply, we do not need divide. With plus, we do not need subtract, with greater than we don't need less than, etc. As we know that if z = x * y, we also know that x = z/y, so by querying the database in the right way, we can find out what z/y is, using the TIMES relationship.

The micro-PROLOG cassette supplied for the Spectrum has not only the interpreter and the SIMPLE front end on, but it also has a further 13 programs, all of which are documented in the micro-PROLOG Reference Manual. Unfortunately this doesn't actually exist yet, but luckily most of the programs are also described in the Primer.

So who is likely to use micro-PROLOG and for what? Well, Logic Programming Associates themselves are currently involved in developing a whole host of example programs, from adventure game designers to teaching aids, and this points to what Sinclair probably see as micro-PROLOG's greatest field of application: education. Simply because the language is so firmly based on logic principles, it is the ideal language with which to teach the ideas of logical thinking. It seems that Logic Programming Associates' prime concern is to teach everyone to stop being 'procedural' when at all possible. In fact, they are almost dogmatic about it, but they are sensible enough to realise that on a fair number of occassions proceduralism (try saying that after half a bottle of logic!) is the only way of thinking.

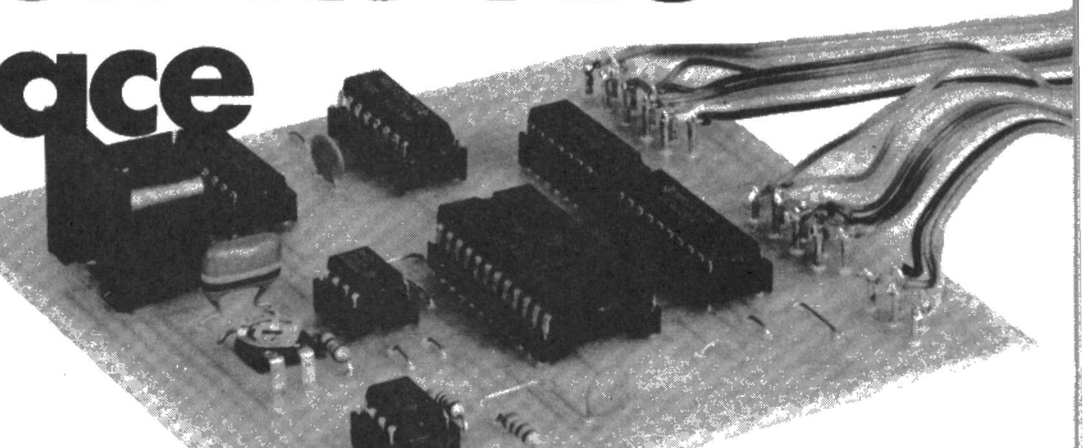
As for the individual, an interest in micro-PROLOG is far more likely to revolve around artificial intelligence applications. It is rather unfortunate then that no provision for anything other than screen and tape I/O has been catered for in the language. This is simply because it is supposed to interface with humans, rather than other machines, but as a version of the interpreter written in C is being developed, those lucky enough to own systems running this language should be able to make their own C based I/O procedures available to micro-PROLOG. Perhaps someone should remind Logic Programming Associates that BCPL is available for the BBC!

"...interest in micro-PROLOG is likely to revolve around AI applications..."

Interface 2 has finally brought ROM-based software to the Spectrum, and along with Interface 1 and the microdrive, a rather respectable micro-PROLOG system could be developed. This is exactly what Logic Programming Associates are doing, but one rather stupid oversight of Sinclair's has caused a lot of trouble. When a ROM cartridge is installed in Interface 2, the microdrive ROM paging system is totally disabled (for information on this paging system, see *E&CM* January, *Understanding 6502 & Z80 Vectors*). This means that they are having to rewrite all the microdrive software and put it in the same ROM as the micro-PROLOG interpreter, which will obviously take a little time.

Nevertheless, micro-PROLOG is a very powerful system that any discerning Spectrum owner should buy. It's the logical choice. ■

Electron RS423 interface



R. A. Penfold links up the Electron to the outside world with a simple and inexpensive communications interface.

To enable the Acorn Electron to operate with equipment such as a modem, printer, or to communicate with another computer, a serial interface is necessary. The type described in this article is an RS423 with signal levels of plus and minus 5 volts. The more common RS232C interface has nominal signal levels of plus and minus 12 volts, but has a minimum requirements of plus and minus 3 volts, and is therefore compatible with the RS423 system.

The circuit has a baud rate of 300, and this is suitable for most purposes, including the majority of modem systems. The baud rate can be changed to 75 under software control, and other rates can be obtained by a simple hardware modification. Eight different word formats can be accommodated, and software is used to select the desired format.

Serial systems

A serial communications link generally has from two to five connecting wires. In its simplest form there is just a connection between the earths of the two computers (or whatever), plus a second wire to carry the data from one piece of equipment to the other, but this permits data to flow in only one direction. If bidirectional communication is required, a third wire is used to carry data back the other way.

Handshaking is sometimes required, as one data terminal might otherwise send data at a faster rate than the other terminal can process it. The handshaking is in the form of a DTR (data terminal ready) or RTS (request to send) output at the receiving end, and a CTS (clear to send) input at the transmitting end. The DTR or RTS line simply signals to the CTS input of the transmitting terminal if data cannot be processed, and a software loop then prevents further data from being transmitted until the CTS input returns to its normal state.

In a bidirectional system, and if handshaking for the data flow in both directions is required, two handshake lines are required, with the DTR/RTS and CTS connections being cross coupled.

Data is transmitted with the least significant bit being sent first, but in addition to the data there is a start bit and either one or two stop bits. Some systems use 8 data bits, but many use only 7 as this is all that the ASCII codes require.

Some systems use a parity bit after the data bits, and there can be either odd or even parity. All this means is that either an odd, or even, number of bits are always transmitted, with the parity bit being added where necessary. The purpose of this is to enable a check for corrupted data to be made at the receiving end of the system. This system is not totally reliable though, and most serial systems do not bother with a parity checking arrangement.

The 6850

This circuit is based on a 6850 ACIA (asyncronous communications interface adaptor) which handles both data transmission and reception. The 6850 contains all the circuitry to deal with start and stop bits, parity checking, etc. Although it cannot handle all of the numerous possible word formats, it can handle most of the more commonly used ones.

Apart from the transmit data output, and receive data input, a CTS input and an RTS output are also provided. The latter can be active when low or high (under software control), and can therefore be used as a DTR output. There is an internal divider circuit for the baud rate clock, and this can be programmed to divide by 1, 16, or 64. In practice a division rate of 16 or 64 is almost invariably used, as in this case, since automatic syncronisation of received data is then provided.

The circuit

Figure 1 shows the full circuit diagram of the interface.

Address decoding is provided by IC1 and IC2. The 74LS20 used in the IC1 position is a dual 4 input NAND gate, but only one of the gates is used in this circuit. This decodes A12 to A15, and gives a low output when all these lines are high. IC2 is a 74LS138 3 to 8 line decoder, and this decodes A8 to A11 (with a negative chip enable input being used to decode A8). The other negative chip enable input is fed from the output if IC1. A negative low output from pin 12 of IC2 is produced when any address in the range &FC00 to &FCFF is addressed. This full address range is unused by internal circuits of the Electron and is available for external in/out circuits. Thus full address decoding is not essential.

As a matter of interest, the BBC microcomputer, from which the Electron was developed, has an RS423 interface provided by a 6805 at addresses &FE08 and &FE09. However, the Electron has no operating software for a 6805 at these addresses, and writing numbers to either of them produces changes in the screen colour!

IC4 is the 6850 ACIA, and this is connected to the data bus via octal transceiver IC3. The latter is needed to provide buffering as IC4 is unable to directly drive the Electron's data bus properly. The read/write terminal of IC4 is connected to the corresponding terminal of the Electron's expansion bus, and this is also coupled to the send/receive terminal of IC3 so that it provides data flow in the appropriate direction. Similarly, the negative chip enable pulses from IC2 are coupled to the corresponding inputs of IC3 and IC4. IC4 also has two positive chip enable inputs, but these are not required here and are simply tied to the positive supply rail.

The 6850 has four registers, but two are read only types and two are write only registers. The device only occupies two addresses, and has a single address input. This is controlled by address line A0. Pin 23 is the data carrier detect (DCD) input, but this is only needed for automatic systems, and in this circuit it is tied to the negative supply rail (the circuit will not operate properly if it is simply left floating).

Pin 14 of IC4 is the Enable input, and it is fed with a timing signal, which is the MPU clock signal in this case. This is not the clock signal that determines the baud rate, and this is set by the clock frequency supplied to pins 3 (receive) and 4 (transmit). It is not normally necessary to have different transmit and receive baud rates, and these two terminals are simply wired together. The clock signal is provided by a straighforward 555 astable (IC5), and VR1 is adjusted to give an operating frequency of 4.8kHZ. When divided by 16 this gives the required 300Hz final clock frequency, or 75Hz can be obtained if IC4 is set for a divide by 64 action. The baud rate is equal to the final clock frequency incidentally. Other baud rates can be provided by altering the value of timing capacitor C3. Changes in value of C3 give an inversely proportional change in frequency (eg. halve the value of C3 to double the baud rate).

The two outputs of IC4 give the standard 0 and +5 logic levels, and the two inputs are designed to operate with the same signal levels. The inputs and outputs therefore require level shifting circuits to provide proper RS423 operation.

IC7 is an MC1489 (or SN75189) quad line receiver, and this is specifically designed for this type of application. As well as providing the level shifting it also has hysteresis which helps to avoid spurious operation due to any noise pick-up in the connecting cable. Apart from level shifting the line receivers also invert the signal to give inputs of the correct polarity to IC4.

An MC1488 line driver was tried as the level shifter/inverter for the two outputs, but this did not operate properly in this circuit. The simple alternative of a dual operational amplifier (IC6) gave better results and is used in the final version of the unit. Each section of IC1 is actually used as a voltage comparator with the non-inverting input biased between the two logic levels by R1 and R2. The inverting inputs are fed from the outputs of IC4. A −5 volt supply is needed for IC6, but this is available from the expansion bus of the Electron. The +5 volt supply is also provided by the Electron.

Construction

Details of the printed circuit board for the interface are provided in **Figure 2.** It is not essential to use IC sockets for most of the integrated circuits, but IC4 is a MOS device, and a (24 pin DIL) IC socket should definitely be used for this device. IC4 should not be fitted to the board until the unit is otherwise complete.

Ideally SK2 should be a printed circuit mounting 5 way DIN socket as it can then be mounted direct on the board, but an ordinary socket of the desired type can be mounted off-board if preferred.

The connection to the expansion bus of the Electron is made via a 23 way piece of ribbon cable about 0.5 metres long. It is unlikely that 23 way cable will be available, but it is not difficult to remove a suitable piece from (say) a piece of 26 or 30 way ribbon cable. One end of the cable connects to the printed circuit board, and this is most easily done if Veropins are used. However, care must then be taken to avoid accidental short circuits, or PVC sleeving must be used over the connections.

The other end of the cable connects to a 2 x 25 way 0.1 edge connector. The Electron's expansion bus has provision for a polarising key, but it may not yet be possible to obtain a matching edge connector. An ordinary 2 x 25 way connector could

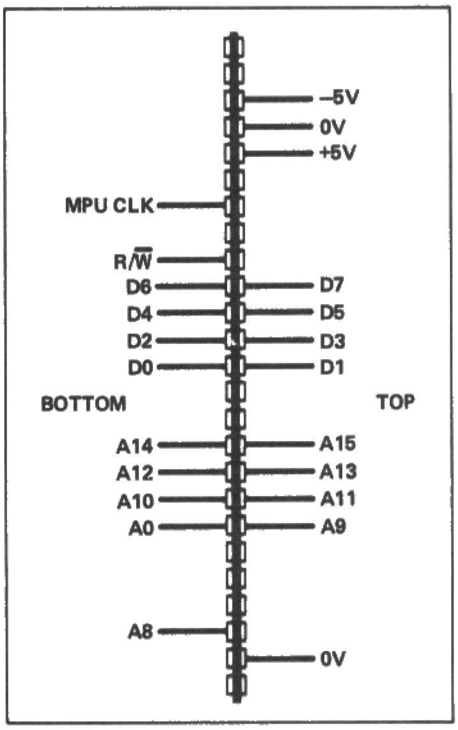
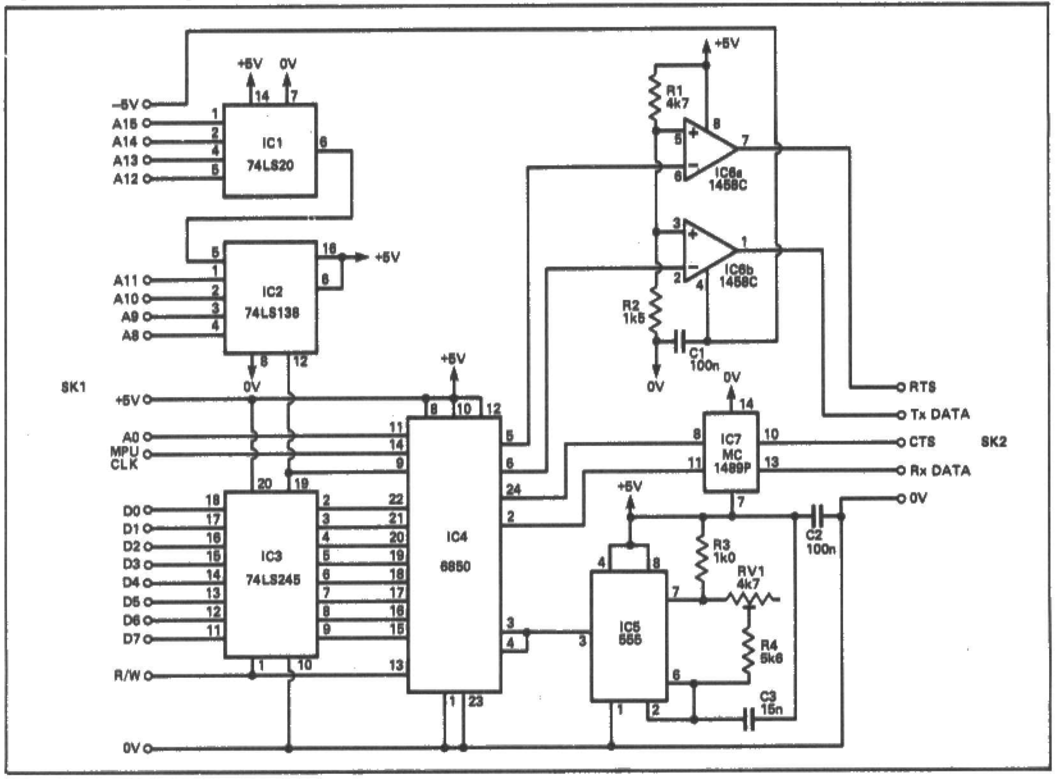
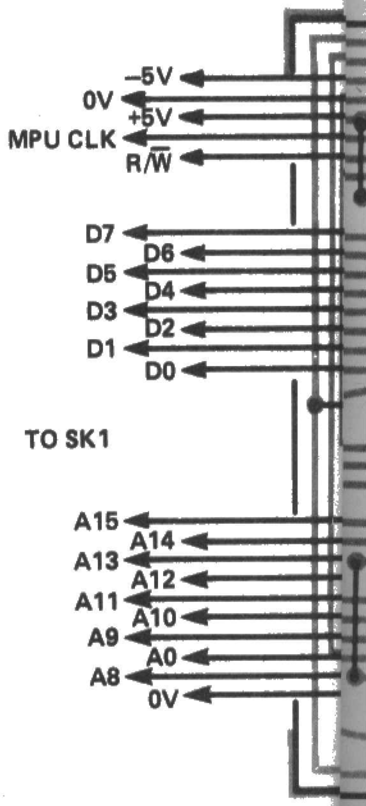
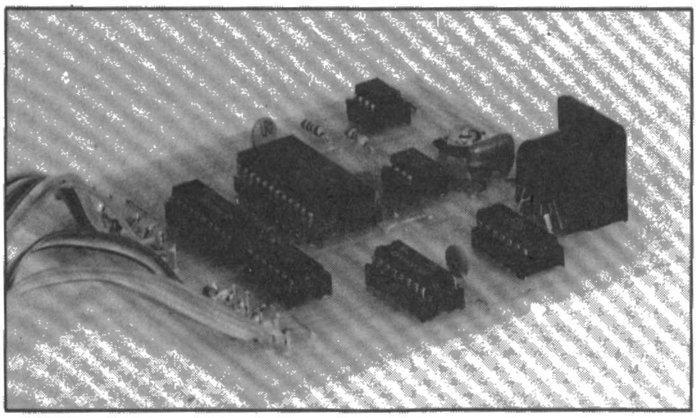


Figure 3. Details of the connections to the edge connector.

Figure 1. Circuit diagram of the Electron RS423 Interface.







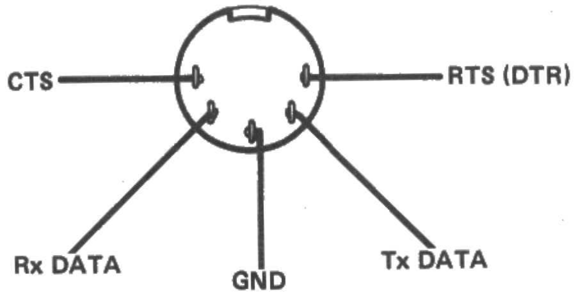


Figure 4. Connections to the In/out socket SK2.

Next month: control registers and software

probably have a polarising key added if a little ingenuity is used, or failing that the top side of the edge connector can be clearly marked as such. **Figure 3** shows connection details for the edge connector (which is shown as viewed looking onto the terminals at the rear of the connector).

Operation

Remember to connect the interface to the Electron before switching on. The computer should then operate normally. Switch off at once and recheck the interface if there is any sign of abnormal operation.

The interface is not difficult to connect to the modem, second computer or whatever, and the way in which the two items of equipment are interconnected was described earlier. **Figure 4** identifies the terminals of SK2, which is shown as seen from the front. When initially checking the unit it is probably best to just wire the TX data and RX data terminals together so that the unit can be tested by transmitting to itself. The simple test program of listing 1 can be tried, and this simply transmits any characters typed into the keyboard, and displays received characters. With the TX data and RX data pins linked, any characters typed from the keyboard should appear on screen.

When transmitting to itself the exact baud rate used is not important, since the transmission and reception rates will always be exactly the same. In normal use the baud rate must be set fairly accurately, and this can be accomplished using a frequency meter to set the frequency at pin 3 of IC5 at 4.8kHz. Alternatively, the unit can be set up for use in a system, and VR1 can then be given any setting that provides reliable operation. ■

PARTS LIST

Resistors (all ¼W 5% carbon)

R1	4k7
R2	1k5
R3	1k
R4	5k6
Potentiometer	
VR1	4k7 0.1W horizontal preset
Capacitors	
C1,2	100nF disc ceramic (2 off)
C3	15nF polyester
Semiconductors	
IC1	74LS20
IC2	74LS138
IC3	74LS245
IC4	6850
IC5	555
IC6	1458C
IC7	MC1489 or SN75189

Miscellaneous

SK1 — 2 x 25 way 0.1 inch edge connector; SK2 — 5 way 180 degree (printed circuit mounting) DIN socket; 24 pin DIL IC socket; 20 pin DIL IC socket; 16 pin DIL IC socket; Two 14 pin DIL IC sockets; Two 8 pin DIL IC sockets; Printer circuit board; Veropins; 23 way ribbon cable; etc.

Figure 2. Positioning of components on the overlay.

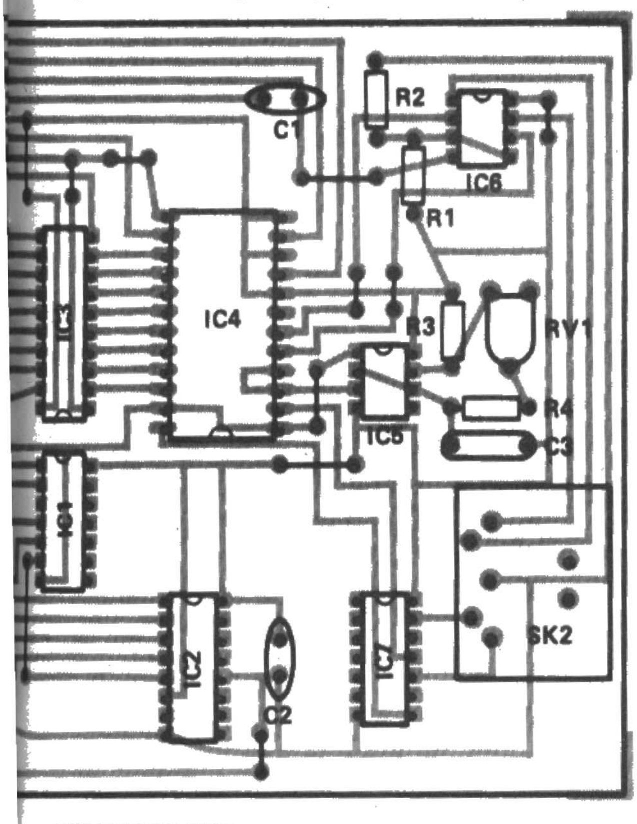
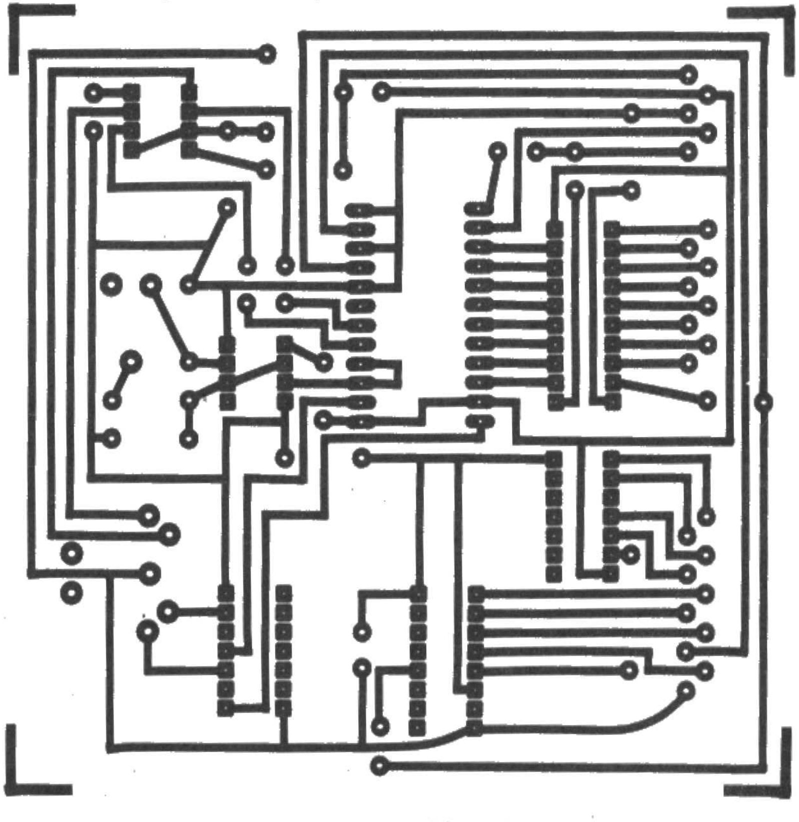


Figure 5. PCB foil pattern.



Micrographic Techniques

Mike James continues his popular series on graphics techniques with a look at how some basic co-ordinate geometry can aid the production of displays.

In last month's Micro Graphics the emphasis was on how screen pixels can be manipulated directly by input devices such as joysticks and light pens. Part of the difficulty of this direct approach is due to the way pixels lack the 'knowledge' to form themselves into the simple geometric shapes that are generally required. If you move a light pen across the screen in roughly a straight line then that is what you will produce — a rough line. The object of the exercise may have been to produce a perfect straight line connecting the two end points but the screen pixels know nothing of this and only respond to the path that the light pen followed! One approach to this problem, smoothing and straightening 'brushes', was described last month but the best and simplest way of drawing a straight line is to get the computer to do it! At one level the idea of using the light pen to indicate two points and then getting the computer to connect them by a straight line is very simple. So simple in fact that many versions of BASIC will draw a line as the result of a single command. However, the general method that lies behind straight line drawing leads to the subject of 'co-ordinate geometry' that many people find quite off putting because of the amount of mathematics that it seems to involve.

Unfortunately if you want to use a computer for graphics there is no choice but to learn a little co-ordinate geometry. This is not as difficult as it sounds because although it does make use of a certain amount of sometimes quite advanced mathematics, the maths always has a very clear physical meaning in terms of what is happening on the screen. This coupled with the use of the computer to work everything out makes co-ordinate geometry a subject within every programmers reach.

Co-ordinates

The idea of using a co-ordinate pair to pick out a single pixel is familiar to almost every programmer. No matter what sort of graphics you are involved in the screen that you are using is a flat two dimensional array of pixels. All drawing on the screen is in terms of selecting pixels using two numbers that measure how far they are from a special pixel called — the origin. The first number measures the horizontal distance and it is called the x co-ordinate and the second number measures the vertical distance and it is called the y co-ordinate. The trouble with this definition is that it fails to specify where the origin is and what units are used to measure the distances. In most cases the origin is located in either the top or bottom lefthand corner of the screen but there is nothing stopping it being located at the centre ar at any other pixel. The way that the distance is measured from the origin also varies from machine to machine. For example, the Dragon and Tandy Colour Computer locate the origin in the top lefthand corner of the screen and measure the y co-ordinate so that increasing y moves you down the screen. On the other hand, the Spectrum locates the origin in the bottom lefthand corner and measures the y co-ordinate so that increasing y moves you up the screen. This variability is ia nuisance in writing graphics programs that are intended for a range of machines.

Another and more subtle problem with screen co-ordinates is that the unit of measurement differs. In most systems changing one of the co-ordinates by one unit moves the position to a new pixel. In other words, the co-ordinates are measured in terms of the size of the pixels and, if the screen co-ordinates vary from 0 to xmax and 0 to ymax, the total number of pixels is:

(xmax+1)*(ymax+1)

This is the simplest scheme but it does become awkward if the machine has more than one graphics resolution. In this case you have to change your graphics programs whenever you change the resolution that you are using. The alternative scheme is to ignore the number of pixels actually on the screen and use a fixed unit to measure co-ordinates. This is the method used by the BBC Micro which, no matter how many pixels there are on the screen uses the same co-ordinate unit which results in the range:

0 to 1280 for the x co-ordinate and
0 to 1024 for the y co-ordinate

Using this fixed system of co-ordinates it is possible to write a graphics program in one resolution and then see the effect it creates in another resolution without changing the program at all! This sounds like an ideal system but it does have a number of very serious problems which have to be taken into account. For example, suppose you are using a resolution that has only 640 pixels horizontally, then there are twice as many integer x co-ordinates as there are pixels. This means that the co-ordinates 0,0 and 1,0 both refer to exactly the same pixel! If you were unaware of this trivial fact you might set the pixel at 0,0 to white and then set the pixel at 1,0 to black and spend a long time staring at the screen trying to work out why you couldn't see the white pixel that you'd plotted! In general, if at the resolution that you are using a single pixel is XINC wide and YINC high (in terms of the co-ordinate system in use) then any detail that involves plotting points that are closer together than this will produce unpredictable results. Going back to the previous example, setting the point x,0 to white and the point x+1,0 to black produces different results depending on the value of x. If x is odd then the points correspond to the same pixel but if x is even then the points correspond to different pixels. Thus, if x is odd you see a single black point but if x is even you see a white and a black point! This is an important effect that can give rise to unstable graphics that change their shape according to where they are positioned on the screen. a more obvious but important problem with having more co-ordinates than pixels is that a lot of time can be wasted plotting the same point more than once.

"...'functional representation' looks to be the key to all graphics problems but..."

Sets of pixels

The fundamental graphics object is the pixel and a single pixel can be specified by quoting its co-ordinates but quoting the co-ordinates of single pixels is not a practical method of specifying the larger objects used in graphics. Somehow we have to find a way of automatically generating the co-ordinates of pixels that form the standard shapes that we use. for example, it is clearly unreasonable to store the co-ordinates of each point on a straight line. What we need is an algorithm that will generate the co-ordinates of each point on the line as needed. Of course there is a well known algorithm for this in the form of the 'equation of a line':

$y=mx+c$

where m and c are constants that govern the angle of the line and its position on the screen respectively. This equation effectively connects values the values of the x and y co-ordinates for points that make up the line. For example, if m=2 and c=3 then the equation is:

$y=2x+3$

and if x is 1 then y=2*1+3 or 6 and the point 1,6 is part of the line. Notice that this only works because for every value of x there is a value of y that gives a point that is part of the line. As x is varied between 0 and XMAX a set of points is generated (not all of which are guaranteed to lie on the screen) that form a straight line.

"Crude, but effective, line editor that is great fun to use".

This idea of using an equation to generate sets of points that make up a curve is easy to generalise. In essence all you need is an equation that connects x and y. This is usually written as:

$y=f(x)$

which means that for a given value of x the function 'f' can be used to calculate a value of y that gives a point x,y that is part of the curve. As another example:

y=SQR(r 2−x 2)

will, if SQR generates the positive square root, pick out all the points that form the upper half of a circle centered on the origin and with radius r. In this case the equation only makes sense for values of x that lie between −r and +r. Outside this range the SQR function is trying to take the square root of a negative quantity and will generate an error.

This way of specifying sets of pixels is known as 'functional representation'. Although functional representation looks as if it might be the key to all graphics problems it is in fact remarkably difficult to find functions which generate any particular curve. For example, you would search a long time before you found the equation that generated the shape of a motor car! In practice there are only three really useful curves that are specified by simple functions and they are — the straight line, the circle and the elipse. These curves can be used in combination to build up larger and more complicated shapes. In many cases computer graphics involves nothing but straight lines! As they are so important the rest of this month's Micro Graphics will concentrate on the use of straight lines to build up outline shapes and the difficult subject of curves in general will be taken up in a future article.

Before moving on, however, it has to be stated that using the functional representation of a straight line is not as easy as it looks if you want to generate an accurate line as quickly as possible. Rather than spend a great deal of time outlining practical line drawing methods it will be assumed that the version of BASIC that you are using will look after the details for you and provides a built-in line drawing command.

Point and line files

Making up shapes out of a collection of lines raises the issue of how the data that defines the line's position, orientation and length should be stored. You could, as many graphics programmers do, choose to ignore the problem and simply store the information in any old way that occurs to you as you write the program. There is a great deal to be gained from thinking out a clear and structured way of storing line data so that it can be easily modified. The obvious way to store a collection of points is as a pair of arrays X(I) and Y(I) used to hold the x and y co-ordinates of each point. This method of holding points is known as a 'point file' and it has the advantage that the index of the arrays also serves to identify each point and making editing possible.

The most obvious way to define a line segment is to give the location of its starting point and its end point. You might think that this implies that the best way to store line information is in four arrays used to store the two pairs of co-ordinates of the starting and finishing points respectively. However, as lines that make up a single shape tend to share start and endpoints it is better to store the co-ordinates of the points in a point file and use the point indices to specify pairs of points. For example, a square would be stored as shown in **Table 1.**

The two arrays X(I) and Y(I) store the co-ordinates of four points 0,0 1,0 1,1 and 0,1 which are the four corners of a square. The two arrays S(I) and E(I) store the start and end points of four lines. For example, the first line starts at point 1 and ends at point 2. In other words, the line is between X(S(1)),Y(S(1)) and X(E(1)),Y(E(1)). In general, the Ith line is drawn between X(S(I)),Y(S(I)) and X(E(I)),Y(E(I)).

This is not the only way to store a list of lines that make up a diagram but it has many advantages that only become clear when you try to change the arrangement of lines so as to make a shape bigger or change its orientation. Changing or transforming shapes is a large subject in its own right and will be treated in next month's Micro Graphics. However, before we can transform images we have to have a way of entering the details of lines and points that form it.

TABLE 1

X(I)	Y(I)	S(I)	E(I)
0	0	1	2
1	0	2	3
1	1	3	4
0	1	4	1

A point and line editor

The program shown is a crude (but surprisingly effective) point and line editor for the BBC Micro which is great fun to use. It allows points to be entered using a joystick for rough positioning and then each point can be 'dragged' into its correct position using the arrow keys. The co-ordinates of the points are kept in two arrays X and Y and P is used to keep a count of the number of points entered. When first run the graphics cursor (a cross) is controlled by the joystick. Pressing "P" enters a point on the screen at the current position of the graphics cursor. To shift the position of any of the points press "C" (for Change) which will change the control of the graphics cursor to the keyboard. In this mode the graphics cursor is positioned over one of the existing points on the screen and moving the graphics cursor updates the co-ordinates stored in X and Y for that point. To move on to another point press "N" (for Next) and the graphics cursor will immediately jump to another point. By pressing "N" repeatedly you can position the graphics cursor over any of the points and drag it to wherever you want. Once you have the points in the correct position you can use the "L" command to join them up.

Importance of position

By positioning the graphics cursor over a point, using the N command any point can be made the start of end of a line. The indices of the points are entered into the arrays S and E in pairs so that the first point that you position the graphics cursor over and press "L" is entered as a start point of the first line, the second point is entered as the end point of the first line and so on. This is the trickiest part of using the program as you have to make sure that you press "L" an even number of times to make sure that for every start point you enter an end point! After you have entered a few lines you can press "D" which will remove the points from the screen and draw the lines so that you can examine the result. If this is not to your liking you can then move any points using the C and N option and redraw the lines — this part is great fun especially if you press the "D" command often!

The program lacks a number of features. In particular there is no way to delete lines or points and no way to save the arrays X,Y,S and E on tape or disc but these options are easy enough to add. To make this task easier, and to facilitate conversion to other dialects of BASIC, the subroutine of **Table 2** is given:

TABLE 2

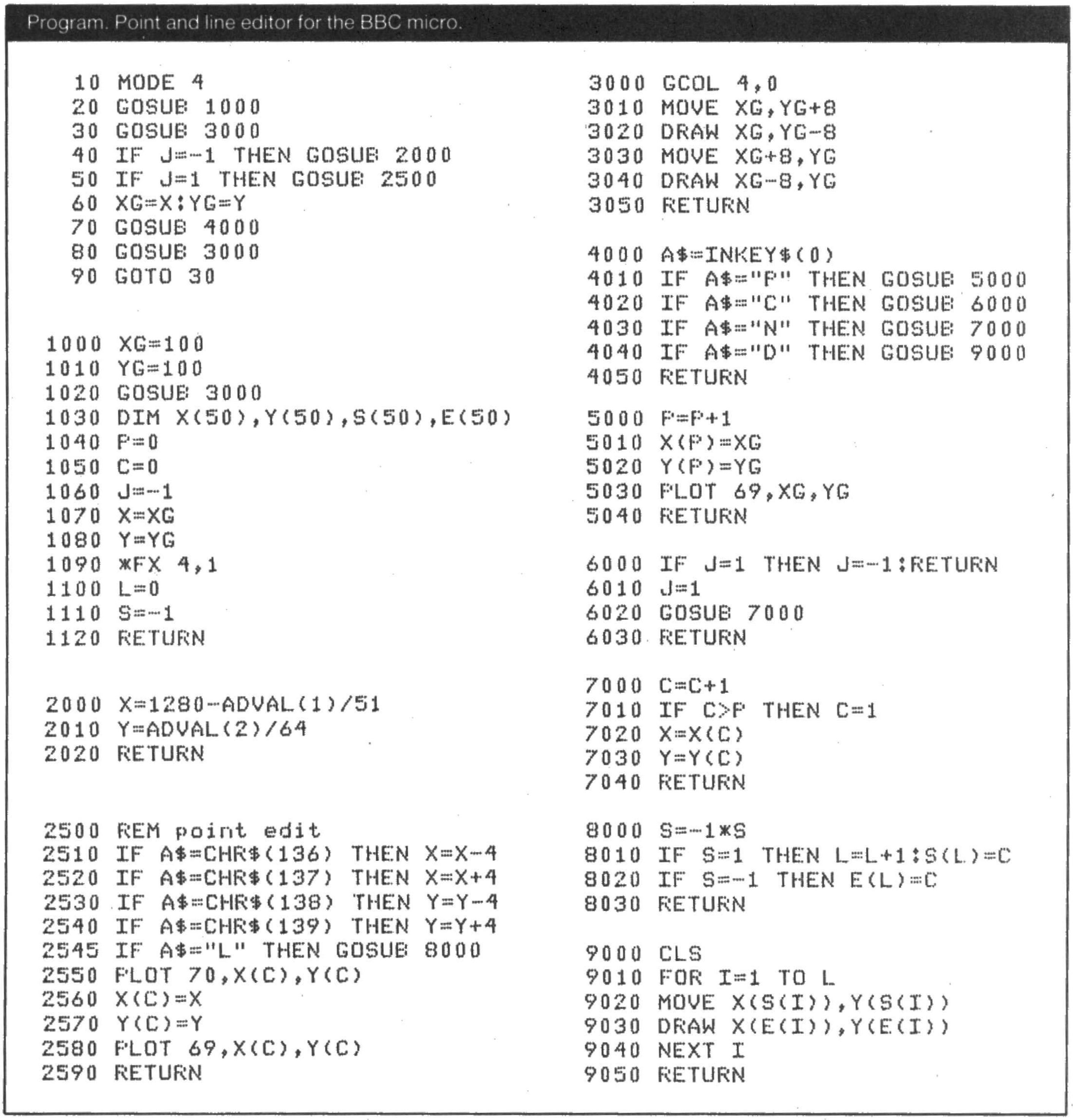
line number	function
10	main program
1000	initialisation
2000	read in current joystick position
2500	keyboard update and point drag
3000	draw graphics cursor
4000	command input and test
5000	P command – add a point to X and Y
6000	C command – 'flip' J to change from joystick to keyboard and vice versa
7000	N command – move C on to next points index
8000	L command – store points index in S or E
9000	D command – draw lines

Conclusion

Using co-ordinates to define shapes raises problems of changing the information that defines the shapes. One way to make this easier is to use a logical method of storing the data based on the co-ordinates of points that make up the start and end points of line segments. The point and line editor given in this month's Micro Graphics will be used next month as a way of building up point and line data that can be used to test and demonstrate a transformation module so it is worth typing it in and becoming familiar with it.

Next Month: Turning things upside down — transformations.

Program. Point and line editor for the BBC micro.



```
  10 MODE 4
  20 GOSUB 1000
  30 GOSUB 3000
  40 IF J=-1 THEN GOSUB 2000
  50 IF J=1 THEN GOSUB 2500
  60 XG=X:YG=Y
  70 GOSUB 4000
  80 GOSUB 3000
  90 GOTO 30


1000 XG=100
1010 YG=100
1020 GOSUB 3000
1030 DIM X(50),Y(50),S(50),E(50)
1040 P=0
1050 C=0
1060 J=-1
1070 X=XG
1080 Y=YG
1090 *FX 4,1
1100 L=0
1110 S=-1
1120 RETURN


2000 X=1280-ADVAL(1)/51
2010 Y=ADVAL(2)/64
2020 RETURN


2500 REM point edit
2510 IF A$=CHR$(136) THEN X=X-4
2520 IF A$=CHR$(137) THEN X=X+4
2530 IF A$=CHR$(138) THEN Y=Y-4
2540 IF A$=CHR$(139) THEN Y=Y+4
2545 IF A$="L" THEN GOSUB 8000
2550 PLOT 70,X(C),Y(C)
2560 X(C)=X
2570 Y(C)=Y
2580 PLOT 69,X(C),Y(C)
2590 RETURN

3000 GCOL 4,0
3010 MOVE XG,YG+8
3020 DRAW XG,YG-8
3030 MOVE XG+8,YG
3040 DRAW XG-8,YG
3050 RETURN

4000 A$=INKEY$(0)
4010 IF A$="P" THEN GOSUB 5000
4020 IF A$="C" THEN GOSUB 6000
4030 IF A$="N" THEN GOSUB 7000
4040 IF A$="D" THEN GOSUB 9000
4050 RETURN

5000 P=P+1
5010 X(P)=XG
5020 Y(P)=YG
5030 PLOT 69,XG,YG
5040 RETURN

6000 IF J=1 THEN J=-1:RETURN
6010 J=1
6020 GOSUB 7000
6030 RETURN

7000 C=C+1
7010 IF C>P THEN C=1
7020 X=X(C)
7030 Y=Y(C)
7040 RETURN

8000 S=-1*S
8010 IF S=1 THEN L=L+1:S(L)=C
8020 IF S=-1 THEN E(L)=C
8030 RETURN

9000 CLS
9010 FOR I=1 TO L
9020 MOVE X(S(I)),Y(S(I))
9030 DRAW X(E(I)),Y(E(I))
9040 NEXT I
9050 RETURN
```

SAT 16

This month our build your own 16 bit computer series, presented in conjuction with Satellite Services, moves on to describe construction procedures.

The SAT-16 system makes use of a number of expensive components and, if these are not to be damaged, it is essential to exercise considerable attention to detail during the computer's construction.

The printed circuit boards of the SAT-16 are double-sided with plated through holes, which makes it quite difficult to remove components once soldered. Therefore, double check that the components are correctly positioned before soldering.

When handling the several HMOS and CMOS devices used on the MPU Card ensure that your soldering iron is properly earthed and use a sheet of aluminium or anti-static foam pad earthed to a radiator or pipe to work on. Ensure that you 'earth yourself' by touching the sheet or pad before handling the ICs.

Most people should have no problem in assembling the cards, but just a word of warning — do not apply pressure to the PCB when soldering as this may cause the tracks to lift and break. Check each joint is good. Note that it is only necessary to solder on the solder side.

It is a good idea to use sockets for all the ICs. Before inserting any chip (especially the 68K) ensure that it is the correct way round and that each pin is correctly aligned (remember that U9 and U11 are only 24-pin chips in 28-pin sockets). Once you are satisfied that all pins are aligned, gently press the chip home, making sure that the pressure is spread over the whole body of the chip and not on one particular point, otherwise the IC body may fracture. This caution applies particularly to the 68000 with its 64 pin DIL package.

Construction begins

SAT-16 MPU assembly — referring to **Figure 1** (PCB overlay) proceed with the assembly as follows:

- Fit and solder all the IC sockets checking that each one is the correct way round and that you do not make any solder bridges between tracks.
- Fit and solder all the resistors and capacitors, checking the polarity of C3, C4.
- Fit and solder diodes, checking the polarity of each one carefully.
- Fit the two edge connectors and before soldering ensure each connector is squarely seated. It may be a little difficult to get the connectors through the PCB but with a little patience it can be done.
- Using insulated wire, fit and solder the jumpers.

Note: At this stage do not install jumpers W3, W4, W11, W12 (A17-A18 to bus) unless the extended G64 bus is being used.

- Install and solder the wire links as follows:

LK1, LK2 baud rate setting, refer to **Table 1.**

LK3 signature routine enable (only installed if using signature analyser, used during card manufacturing and testing, normally removed)

LK4 68K diagnostic enable (only used if user includes start up diagnostic routines in 68K software, normally disabled by installing link)

LK5, LK6 RAM selection, selects which RAM type is being used (see **Table 2**).

- Fit and solder the two crystals.
- Before installing the integrated circuits the following resistance check should be carried out:

Resistance Check — using a meter check the resistance between the power and ground lines (pins 31/32 on 64-way edge connector). If the meter shows a short (ie zero ohms) check the soldering until the problem is located.

Power/ground line check — using a meter check that each IC is connected to both the power and ground lines.

- Once satisfied, install the integrated circuits, ensuring that they are the correct way round, leaving the CMOS and HMOS devices until last. Next, insert the 6116 RAMs making sure that they are installed in the lower portion of sockets U11, U9. The monitor EPROMs (if required) 2950 and PALs should then be installed, finally insert the 68000 and 68701 processors.

Assembly is now complete but carefully double-check completed board before connecting to power supply.

VDU/printer adaptor

Different adaptors are used to allow different types of I/O devices to be connected to the SAT-16 MPU parallel data port. The assembly of the standard adaptor allows a VDU and printer to be connected. Compared to the MPU the adaptor is fairly straightforward, referring to **Figure 2** (PCB overlay) proceed as follows:

- Fit and solder the four IC sockets.
- Fit and solder all the resistors and capacitors, checking the polarity of C1
- Fit and solder the 40-way Ansley edge connector.
- Using spaces and screws, fit and solder the cannon and amphenol connectors onto the board.
- Links LK1-LK4 are used to set the polarity of the STROBE and BUSY lines for the printer; check and set up according to whichever printer is being used. **Note:** the MPU Card is configured as a slave device. If you wish to change it to a master then cross over the TXD and RXD lines on the RS232 interface.
- Connect the Reset Switch using flying leads to the pins (P1, P2 and P3 — ground).
- Check resistance between power and ground lines as before. If all is OK insert ICs 1-4.

The adaptor is now complete, but once again double-check complete board before connecting power.

TABLE 1

LK2	LK1	Baud Rate
0	0	38.4K
0	1	4800
1	0	500
1	1	130

TABLE 2

LK6	LK5	Mem. Select
0	0	Invalid
0	1	5564 or eq.
1	0	6116 or eq.
1	1	Invalid

TABLE 3

PIN 31a/b	+5V	2 amps
PIN 30b	−12V	200 ma
PIN 30a	+12V	200 ma
PIN 32a/b	GROUND	GROUND

Note: If using the extended G64 bus, then 1a/1b are connected to address lines A17/A18.

AMPHENOL 36-way (J3)

Pin No.	Signal
1	GND
2	not used
3	RXD
4	not used
5	TXD
6	not used
7	RTS
9	not used
9	CTS
10	not used
11	DSR
12	not used
13	GND
14	DTR
15	$\overline{\text{RESET}}$
16	not used
17	reserved
18	not used
19	reserved
20	not used
21	H2
22-26	not used
27	H1
28	PRSTRB
29	PRD0
30	PRD1
31	PRD2
32	PRD3
33	PRD4
34	PRD5
35	PRD6
36	$\overline{\text{CRASH}}$
37	$\overline{\text{RST}}$
38	VCC
39	RST
40	GND

ANSLEY 40-way (J1)

Pin No.	Signal
1	GND
2	RXD
3	TXD
4	RTS
5	CTS
6	DSR
7	GND
8-19	not used
20	DTR
1-25	not used

CANNON D-TYPE (J2)

Note: port is configured as DTE

Pin No.	Signal
1	PRSTRB
2	PRD0
3	PRD1
4	PRD2
5	PRD3
6	PRD4
7	PRD5
8	PRD6
9	not used
10	ACK
11	BUSY
12-18	not used
19-29	GND

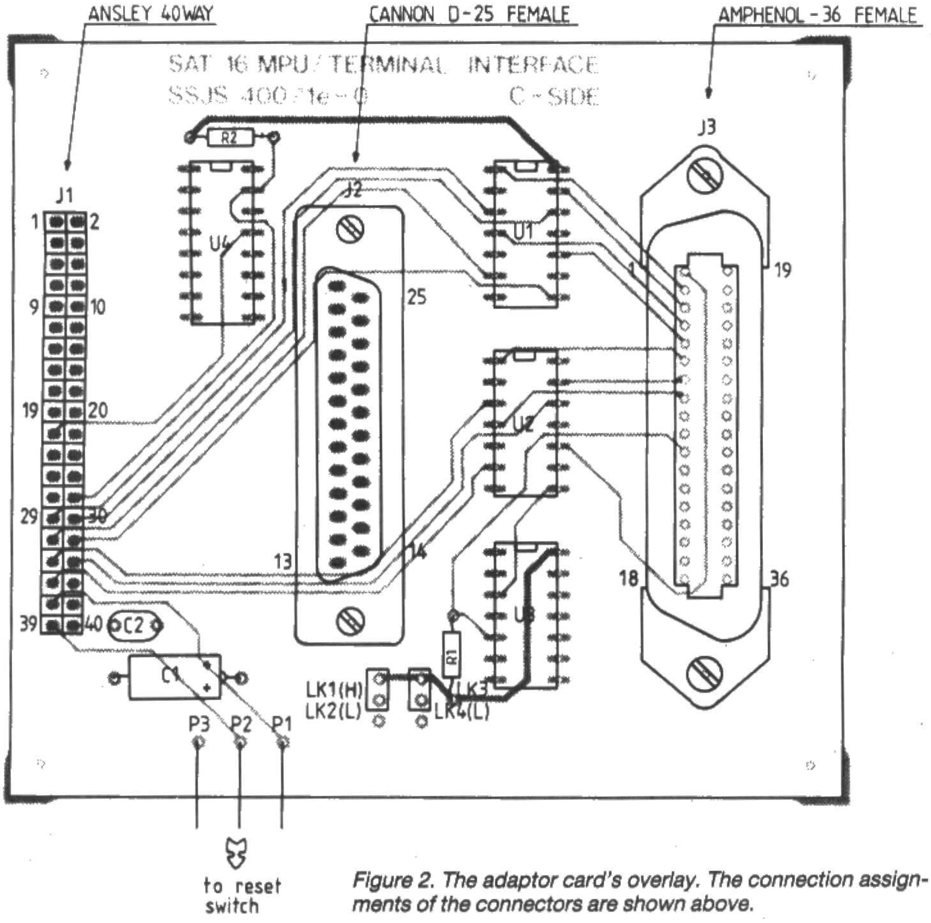
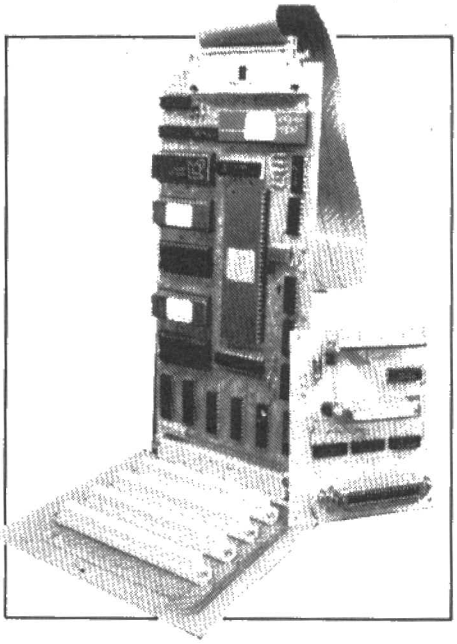


Figure 2. The adaptor card's overlay. The connection assignments of the connectors are shown above.



Power Connections

The SAT-16 MPU Card requires the following supplies as a minimum. It is recommended, however, that a minimum of 9 amps be made available on the +5V to allow for future expansion.

Connection and Initial Test

Once satisfied that you have checked the Cards thoroughly, proceed as follows:

- Connect the adaptor to the 40-way Ansley socket (J2) on the MPU Card. If the MPU Card is being used as a stand-alone card then AS should be linked back to BERR on the connector (J1).
- Connect the +5V, +12V and −12V lines to the connector J1. (It is recommended that either the "mini" backplane be used or a 64-way female DIN connector).
- Switch on the units; if all is satisfactory the following messages should appear on the VDU screen:

```
SAT 16 MPU V2.1
TIME HHMM?
```

A "printer not found" message also appears if the printer is missing. The system will then wait for the time prompt to be answered. Once this has been done the system will respond by outputting the time so that a check can be made. At this point no further progress can be made until the 68000 is loaded with suitable software — either the monitor (SATBUG) or user software.

If when powered on nothing appears on the screen, switch off at once and check the Adaptor and MPU Card thoroughly until the fault is found. Remember also that the VDU must be set to the correct baud rate and that both send and receive rates are the same.

A complete kit of parts for both boards is available from Satellite Services.

▶

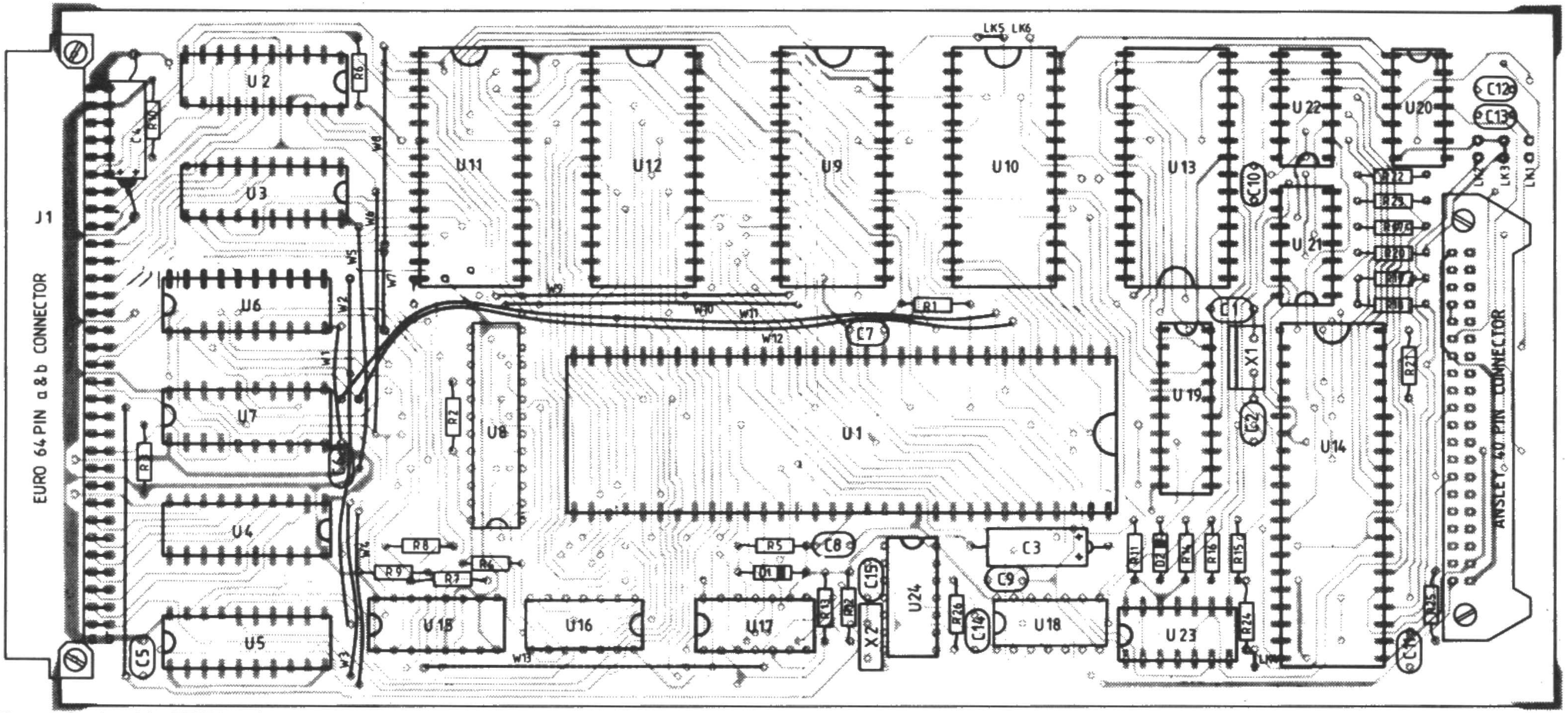


Figure 1. Overlay of the 68000 board.

PARTS LIST

Integrated Circuits

IC1	68000
IC2,3	8303
IC4,5	LS245
IC6,7	LS244
IC8	PAL20L10
IC9,11	RAM 6116 or 5564
IC10,12	EPROM 2732 or 2764
IC13	AM2950
IC14	68701
IC15	LS348
IC16	7407
IC17	LS24 or LS00
IC18	LS14
IC19	PAL16L8
IC20	LS00
IC21	1489
IC22	1488
IC23	LS20
IC24	LS04

Resistors (all ¼W)

R1,2,3,4,5,6,7,8,9,10,11,12,13,18,19, 20,21,24,25	4k7 or 6k8
R5	47k
R17	220R
R14,15,16	10k
R22,23	27k

Capacitors

C1, C2	10 or 12p
C3, C4	100u
C5, C11	10n
C12, C13	47n

Diodes

D1, D2	IN4148

Connectors

J1	64-way a & b Euro connector
J2	Ansley 40-way

Miscellaneous

XTAL1 2.5MHz; XTAL2 8.0MHz; Link wire; Connector screws; IC sockets.

Adaptor

Semiconductors

IC1,2	SN74128
IC3	SN74LS86
IC4	SN74LS367

Resistors

R1	10k
R2	4k7

Capacitors

C1	10u
C2	10n

Connectors

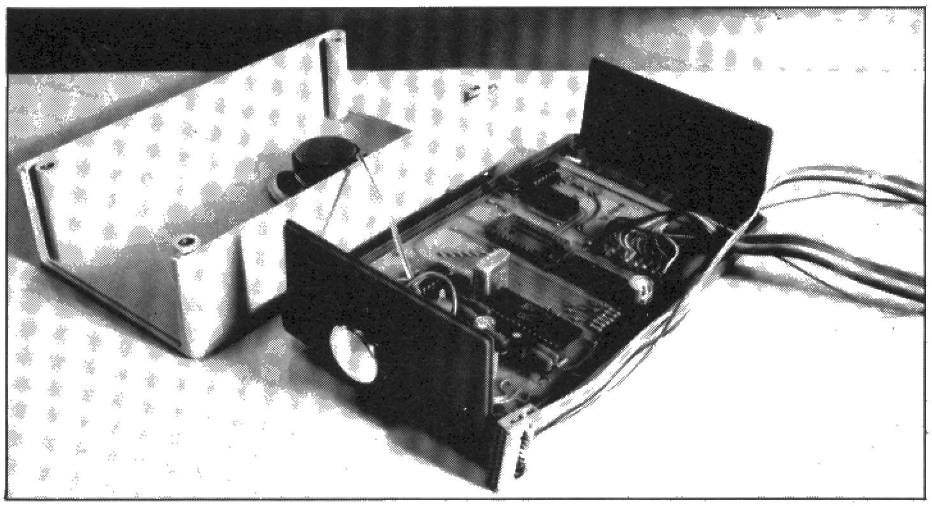
Ansley 40-way socket
Cannon D-type 25-way female
Ampenol 36-way female

Miscellaneous

Link posts; push-button.

Speech synthesiser for BBC micro

And now a speech synthesiser for the BBC micro. Mark Stewart builds a low cost board with the same SPO256-AL2 allophone chip used in last month's project for the Spectrum, but with the added extra of EPROM storage.



Following on from the discussion of speech synthesis in the November and December '83 issues of *E&CM*, this article presents a practical speech synthesiser for use with the BBC Model B. Using the General Instruments allophone IC, the SPO256-AL2, this circuit allows two different modes of operation. In "direct" mode the computer controls the speech synthesiser IC directly, allowing total software control of individual allophones, which may be strung together to produce the required speech. In the second mode the computer selects a word or phrase from an on-board EPROM; 64 phrases, each up to 32 allophones long can be stored in a single 2716 2K EPROM. Alternative connections allow the format to be changed so that 32 phrases of 64 allophones, or 16 phrases of 128 allophones can be used.

It is also possible to use a 2732 4K EPROM and select between 2 banks of 2K using an on board switch. Switching between the two modes is done in software, so that a program can mix "standard phrases" from the EPROM with direct, software controlled words.

A particular advantage of the EPROM mode is that the computer is freed once it has initiated the phrase required. Thus fast action games can be combined with speech, using simple programming, without loss of speed. It is of course, necessary to construct a speech EPROM vocabulary for a particular set of programs. The "direct mode" allows the words or phrases to be set up experimentally in RAM before committing them to EPROM, using an EPROM programmer (but of course, the system can be used without an EPROM).

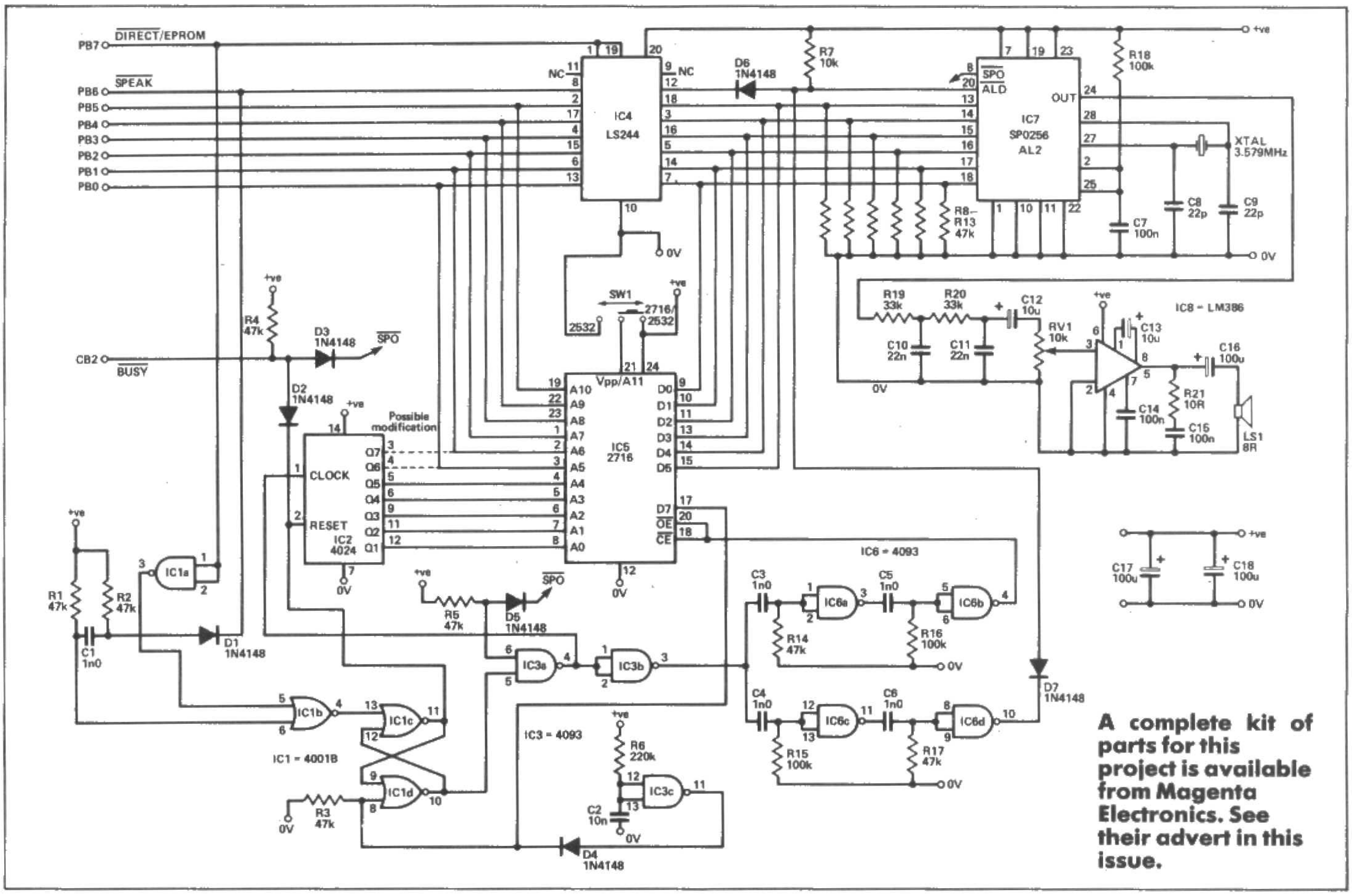
The Circuit

A full circuit diagram is shown in **Figure 1.** The speech synthesiser IC7 has 6 data input lines which address the 64 allophones listed in **Figure 4.** The direct mode is selected by a logic low on the computer user port line, PB7. Tri state buffer, IC4, is enabled and the 6 data lines of IC7 are connected to PB0-5 of the computer user port. User port line PB6 is IC7. A signal from the active low output (SPO) of IC7 is coupled to the CB2 control line of the computer port. This line is pulled low by IC7 when it is busy. A rising edge indicates to the computer that the next allophone data can be loaded.

Selection of direct mode automatically disables the EPROM. Its outputs assume a high impedance state and are effectively disconnected from the system.

When PB7 is high EPROM speech is enabled. IC4 output lines are all set to their high impedance state. IC1a inverts the signal from PB7 and sets one input of IC1b low. The other input of NOR gate IC1b is fed from the SPEAK line PB6 via D1 and the pulse forming network R1, R2 and C1. A high to low transition of PB6 produces a short positive pulse at the output of IC1b. This pulse sets the bistable formed by cross coupled gates IC1c and d. The output of IC1c changes from high to low. This removes the reset condition from the 7 bit binary counter IC2, and pulls down the BUSY line via D2. Setting the bistable also changes the output of IC1d from low to high. The output from IC1d puts a high state on one input of IC3a, the other input of which is held high by the inactive SPO output of IC7. The output of IC3a therefore changes from high to low, and IC2 is clocked 1 count from zero. The output from IC3a is inverted by IC3b and coupled to the two delayed pulse forming circuits IC6a, b and IC6c, d. The low to high transition from IC3b is coupled, via C3, to the inputs of IC6a. The inputs of IC6a are pulled high via C3. C3 then begins to charge via R14, and the inputs are pulled back down again to zero, even though the output of IC3b may still be high. The result of this is that a short negative pulse is produced on the output of IC6a, starting when IC3b output changes from low to high. The length of the pulse is set by the values of C3 and R14.

IC6b is an identical stage which produces a negative pulse which starts on the rising edge at the end of the negative pulse from IC6a. A similar pattern occurs with IC6c and IC6d, but the relative pulse lengths differ because different value timing resistors are used. **Figure 2** shows these pulses graphically. The output pulse from IC6b enables the EPROM. The falling edge of the pulse on the output of IC6d is timed to occur during the time that the EPROM is enabled. This pulse is used by IC7 which loads the data present on the EPROM data lines, and starts to execute the first allophone routine. As soon as IC7 loads the data the 'busy' condition is signalled by the SPO output changing from high to low. This signal is coupled via D5 to IC3a. The inputs to NAND gate, IC3a, change from both being high to one being high and one being low. Its output changes from low to high. This change has no effect on the counter, which is clocked on falling edges. The change is inverted by IC3b, but has no effect on IC6 which is triggered only by rising edges.



A complete kit of parts for this project is available from Magenta Electronics. See their advert in this issue.

Figure 1. Circuit diagram of the synthesiser.

The circuit waits in a stable state until IC7 has completed its routine. At the end of the routine the SPO output changes from low to high, IC3a output also falls and clocks IC2 and IC6 repeats its pulse sequence. IC7 again loads the EPROM address, which is derived from IC2, and has increased by 1. IC7 therefore executes the allophones from the EPROM one by one. With IC2 and IC5 connected as shown, the lower 5 address lines would cycle indefinitely without some means of ending the sequence. To accomplish this, one of the two higher order data lines of the EPROM is used. The 64 available allophones require only 6 data lines from the 8 bit EPROM. The higher two lines are normally set to 0. By setting one of the high order lines to logic 1, it is simple to indicate the end of a sequence. EPROM data line D7 is connected ot IC1d. The control bistable is reset whenever a logic 1 appears on D7, IC2 is reset, and the BUSY signal returns from low to high. The circuit is then returned to its initial state.

"... allows the computer to select words or phrases stored in EPROM".

A 'power on reset' pulse is produced by IC3c to ensure that the control bistable is initially in the correct state.

IC7 has its own clock oscillator which uses C8, C9 and crystal X1. A 3.12MHz crystal is specified by the manufacturers, but a cheaper 3.57MHz USA colour TV crystal operates well. The speech output from IC7 passes through a simple low pass filter network and volume control, to a standard IC audio amplifier. A miniature speaker provides the final link.

Construction

Construction is very straightforward. A double sided printed circuit board is used, see **Figures 3a** and **3b;** the only off board components are the volume control RV1 and the loudspeaker.

Follow the overlay drawing to be published next month when assembling the board. Note that through connections are made using PCB pins, and that some component leads are soldered on both sides of the board. IC sockets are recommended but are not essential except for the EPROM.

Connections to the speaker, RV1, and the ribbon cable from the computer are best made using PCB pins.

Although only 11 connections are made to the computer it is better to use a length of 20-way ribbon cable, and to cut back the unused leads at the synthesiser end.

Programming

Direct mode allows the computer to operate the speech synthesiser IC directly. The mode is selected by setting the user port line PB7 to a logic 0. The user port must first be set up as an output port by using the instruction:

?&FE62=FF

PB7 is set low by any number less than 127. The production of an allophone is initiated by a high to low transition of PB6. The other 6 port lines select the required allophone. The allophone table, **Figure 4,**

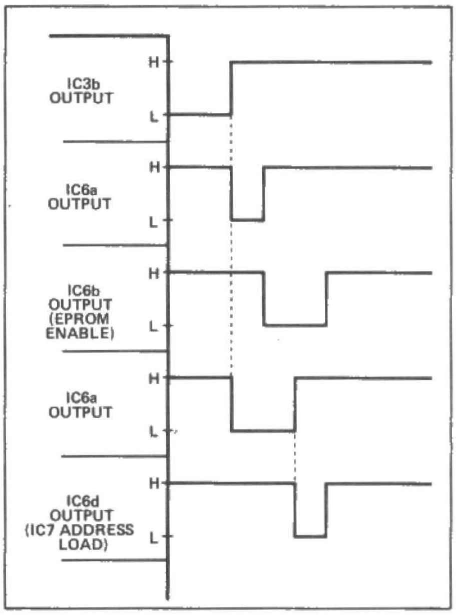


Figure 2. The relationship between the various timing pulses.

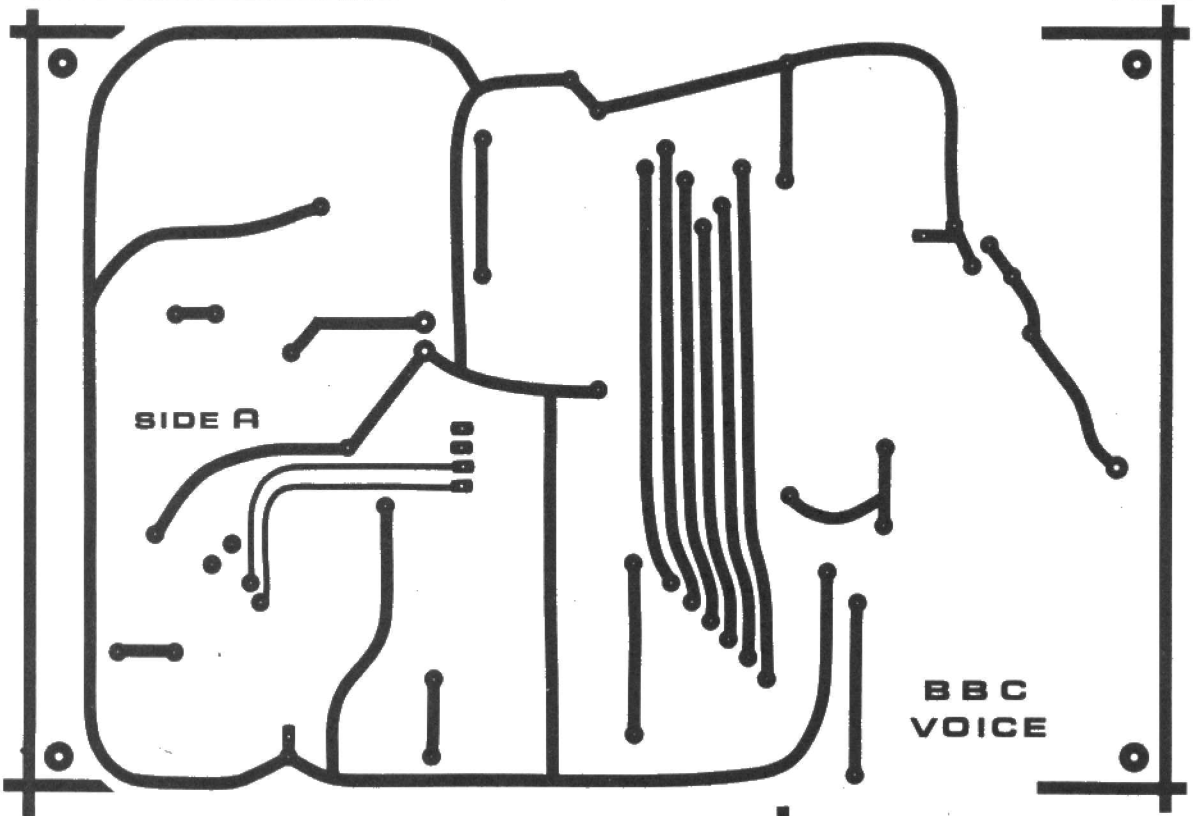


Figure 3a. Top side foil of the double sided board.

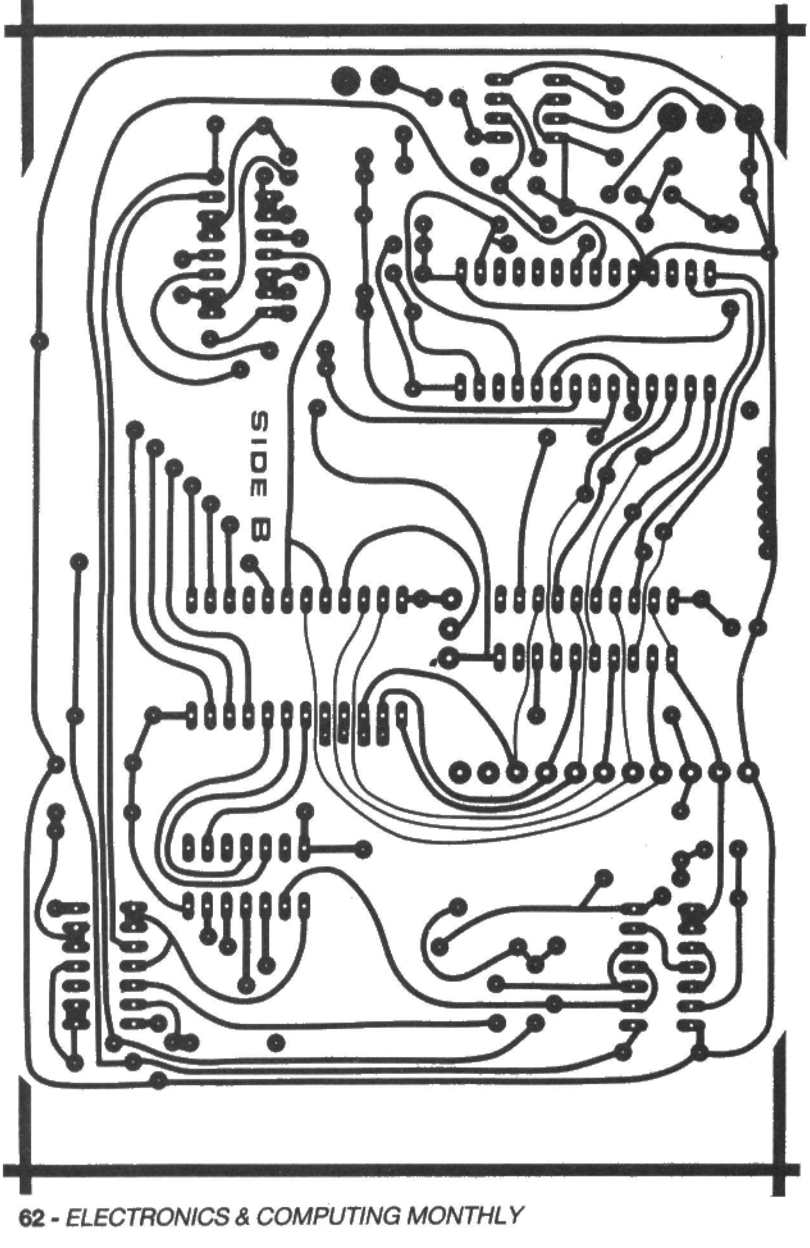


Figure 3b. Lower foil pattern.

lists the 64 sounds and pauses that can be produced.

The following short program allows each allophone to be selected and sounded:

```
10  ?&FE60=64
15  INPUT A
20  ?&FE60=A
30  GOTO 10
```

Line 10 sets all the port lines except PB6 to 0. The decimal number of the required allophone is entered in line 15. In line 20 the allophone number is put on port lines PB0-PB5, and PB6 is changed from high to low. The falling edge on PB6 tells IC7 to latch the allophone data into its register, and commence execution. There is a problem with this simple program because IC7 continues to produce the allophone until a new one is entered. A pause (data 0-4) must always be entered to finish a phrase or word.

It is necessary for the computer to know when one allophone has ended, so that the next one can be loaded inot IC7 without leaving an audible pause. This timing function is made possible by the SPO output of IC7 which goes low whilst each allophone is sounded. A low to high transition on the SPO line occurs at the end of each allophone. This BUSY signal is coupled to the computer port control line CB2.

Those familiar with the 6522 computer VIA chip will be aware of its many complexities. Without going into detail the relevant operation can be summarised as follows. Control line CB2 influences data Bit 3 of a register in the 6522 known as the interrupt flag register (IFR). The register can be read by the computer at FE6D Hex. Normally the register contains all '0s', so a simple test can be used to check on Bit 3.

The effect of CB2 on IFR Bit 3 is controlled by another 6522 register at FE6C called the peripheral control register (PCR). By setting Bit 6 of the PCR to a logic 1, a low to high transition on CB2 will set Bit 3 in the IFR to 1. Three of the bits in the PCR are already set (Bits 1, 2 and 3) and this must be allowed for when setting Bit 6.

When the computer sends an output to the user port PB lines Bit 3 in the IFR is automatically set to 0. After IC7 has executed the appropriate allophone, the BUSY line switches from logic 0 to logic 1 so instructing the computer to issue the next instruction.

The simple BASIC program that follows should clarify the method of operation:

```
   4 ?&FE62=&FF                  Sets up the output port
   8 ?&FE6C=78                   Sets up the PCR
  10 ?&FE60=64                   Sets PB6 high
  20 READ A                      Get allophone data
  30 IF A=64 THEN GOTO 120       Tests for end condition
  40 ?&FE60=A                    Loads data into IC7
  50 B=?&FE6D                    Reads IFR
  60 IF B=0 THEN GOTO 50         Loop while allophone is executed
  70 GOTO 10                     Repeat for next allophone
  80 DATA 20,29,36,47...etc..64  List of allophone codes separated by commas
                                 ending with 64
 120 STOP
```

The program loops around lines 50 and 60 whilst the BUSY line is low. When the BUSY line is switched to high, B is no longer 0 so the program proceeds to line 70, and returns for the next.

EPROM Programming

The EPROM mode of operation is set by a logic 1 on PB7 of the computer port. Speech is initiated by a high to low transition of PB6. The other 6 data lines determine which bank of 32 EPROM addresses is used. If PB 0-5 are set to zero the EPROM is addressed from 01 to 32. If PB0 is set to 1 the 32 EPROM addresses start at 33 and end at 64, etc.

Note that the sequence can be made shorter than 32 bytes by putting an end character in the EPROM (any number between decimal 128 and 255, or 80-FF Hex).

All of this translates into BBC code as follows:

First enter: ?&FE62=FF

This sets the user port lines PB 0-7 to output mode.

Then enter the following short program:

```
 5 INPUT A
10 ?&FE62=192+A
20 ?&FE60=128+A
30 GOTO 5
```

The value of 'A' determines the bank of addresses to be used, and can take a value from 0 to 63.

Adding 192 to A in line 10 sets PB6 and 7 high, and so sets up EPROM mode. Line 20 leaves the value of PB7 high whilst changing PB6 from high to low. The falling edge of PB6 initiates the speech cycle. Note that if an end character is not found in the EPROM the circuit will latch, and continuously cycle through the 32 addresses. Switching the power off and on will restore normal operation.

Alternative phrase lengths can be set by changing the hardware connections. If the Q6 output of IC2 is connected to A5 of IC5, and the computer PB0 line disconnected, there will be 32 banks of 64 addresses available. Linking Q7 of IC2 to A6 of IC6 as well, will give 16 banks of 128 addresses.

To use the speech in a program it is only necessary to run lines 10 and 20 with the required value of 'A' added to the 192 and 128. Remember though to initiate the port by entering ?&FE62=FF somewhere in the program, before the speech part.

Mixing EPROM speech with direct speech can be achieved simply by switching backwards and forwards using software. The busy condition may be read from the IFR in exactly the same way for both the direct and EPROM modes. ■

PARTS LIST

Resistors (¼Watt 5% Carbon Film)	
R1,2,3,4,5,8,9,10,11,12,13,14,17	47K
R6	220K
R7	10K
R15,16,18	100K
R19,20	33K
R21	10R
RV1	10K log potentiometer
Capacitors	
C1,3,4,5,6	1nF ceramic plate
C2	10nF C280
C7,14,15	100nF
C8,9	22pF ceramic plate
C10,11	22nF C280
C12,13	10uF 10V radial electrolytic
C16,17,18	100uF 10V radial electrolytic
Semiconductors	
D1-7	IN4148
IC1	4001B
IC2	4024B
IC3,6	4093B
IC4	74LS244
IC5 optional	2716/2732 EPROM
IC7	SPO256-AL2
IC8	LM386

Miscellaneous

Crystal - X1 3.579MHz or 3.12MHz; Speaker - Miniature 8 ohm; 11 way ribbon cable; IC sockets; Socket for micro port; Knob; Case; Fixings; PCB etc.

Figure 4. The 64 allophones generated by the synthesiser.

ADDRESS	ALLOPHONE	EXAMPLE WORD	ADDRESS	ALLOPHONE	EXAMPLE WORD	ADDRESS	ALLOPHONE	EXAMPLE WORD
0	10ms PAUSE		22	UW1	tO	44	NG	aNchore
1	30ms PAUSE		23	AO	AUght	45	LL	Lake
2	50ms PAUSE		24	AA	hOt	46	WW	Wool
3	100ms PAUSE		25	YY2	Yes	47	XR	repaiR
4	200ms PAUSE		26	AE	hAt	48	WH	WHile
5	OY	bOY	27	HH1	He	49	YY1	Yes
6	AY	skY	28	BB1	daB	50	CH	CHurch
7	EH	End	29	TH	THin	51	ER1	summER
8	KK3	Comb	30	UH	bOOk	52	ER2	bURn
9	PP	Pit	31	UW2	fOOd	53	OW	nOW
10	JH	dodGe	32	AW	OUt	54	DH2	THey
11	NN1	thiN	33	DD2	Do	55	SS	veST
12	IH	sIt	34	GG3	wiG	56	NN2	No
13	TT2	To	35	VV	Vest	57	HH2	Noe
14	RR1	Rural	36	EF1	GUest	58	OR	stORe
15	AX	sUcceed	37	SH	SHip	59	AR	alArm
16	MM	Milk	38	ZH	aZure	60	YR	cleaR
17	TT1	parT	39	RR2	bRain	61	EG2	Got
18	DH1	THey	40	FF	Food	62	EL	saddle
19	IY	sEE	41	KK2	sKy	63	BB2	Business
20	EY	bEIge	42	KK1	Can't			
21	DD1	coulD	43	ZZ	Zoo			

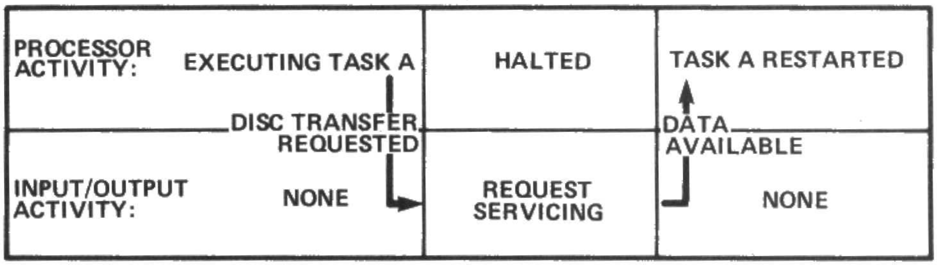
MULTIPROGRAMMING

Networking and timesharing are practices which are gradually working their way down from industry into home computing. Meanwhile the arrival of a second processor for the BBC opens up the possibility of multiprocessing. James Dick examines ways of improving computer efficiency in the first of a two part series.

Multiprocessing systems are so common in nature that it is, perhaps, surprising that they are comparatively rare in computing. The principle of multiprocessing is evident in any animal colony where the principal task — ensuring the survival and advancement of the colony — is shared among the individuals; each member of the colony has considerable autonomy in day-to-day tasks but is bound by species-wide behavioural patterns which may be instinctive or agreed upon by consensus. The translation of this model into computing hardware is fairly straightforward; the main task being run by the computer system is shared among many processors.

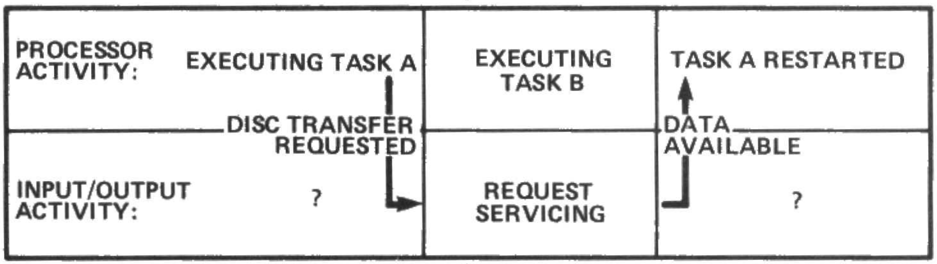
This 'sharing' technique is known as multiprocessing. It has considerable advantages compared to monoprocessor systems in terms of total throughput and is becoming increasingly used as the cost of the processor itself continues to decrease. Before discussing true multiprocessing, multiprogramming is considered as an introduction to some of the problems encountered.

Multiprogramming techniques enable a single processor to tackle several tasks in an interleaved fashion. Consider a single processor executing a task — the processor will require access to bulk storage media (disk, tape) and, perhaps, a printer. When access is requested, a finite time interval will occur before the data requested becomes available to the processor. During this time the processor will have been idle — and even a delay of a few milliseconds means thousands of instructions which might have been executed have been delayed **(Figure 1).** Multiprogramming, however, allows a "time-sharing" system to operate and offers partial solution to the problem **(Figure 2).** Now, while the processor is waiting on the data transfer, it proceeds to execute anothet task.



▲ *Figure 1. Single task execution.*

Figure 2. Dual task execution. ▼



> **"... these techniques enable a single processor to tackle several tasks ..."**

The advantage of time-sharing can be seen at any computer centre where many users may use the same computer simultaneously and suffer only minor delays in the system's response to their commands. It should be noted that while multiprogramming increases the throughput of any system, the prime reason for its implementation is to save the valuable time of the users; from their viewpoint, waiting on the machine finishing other tasks in a batch queue before processing their task is a feature of the past.

A simplified explanation of the multiprogramming mechanism is as follows. The computer's operating system runs a task called the supervisor whose function is to co-ordinate input/output, the execution of users' tasks, and system functions. When a user logs-in to the system, the supervisor sets up a table in memory containing the user's account number, privileges, and devices allocated (either on a shared basis or single). This table also stores information that is vital when that user's task is halted — the return address and processor status before the halt. For example, if there are two users (A and B) on the system, A's task may have requested access to data held on disk. The processor passes this request to the disk controller and halts A's table. B's task is then restarted by filling the processor's registers with the information held in B's table. Some time later, the disk controller interrupts the processor as the data for A's task is ready for transfer. The processor stores its registers into B's table (rather like a very complex subroutine or interrupt call in a microprocessor) so halting B's task, and then locates the cause of the interrupt, determines which task needs restarting (A's) and transfers the data **(Figure 3).**

The basic system described above implies that tasks making extensive use of input/output may slow the system down for the remainder of the users and, for a variety of reasons, this is the case with timesharing systems. Another timesharing philosophy which may also be employed is round-robin scheduling; the processor's time is split into small units (few tens of milliseconds) and these are shared out between users.

Hence, multiprogramming requires a supervisor program (which may be called a monitor, executive, or operating system) and a complex interrupt request interface between the users and peripherals, and the supervisor. The supervisor, in its role as system controller must also provide memory protection between tasks and itself, and update the users' tables.

If there are a large number of users, it would be very wasteful to keep their tasks in core memory when the tasks were not executing. Indeed, it is wasteful of space to keep any part of a program which is not actually being processed in core memory. If the system has fast links between core and disk then only the program segments about to be executed need be in core: as more program is required, it is transferred from disk to. core "on demand". This technique of cheating the processor into believing it has more core than real core is called paged virtual memory; keeping trace of a task's program location is another duty of the supervisor. Virtual memory is divided into pages, each containing a fixed number of memory locations; the virtual address of a variable, for example, is made up of a page number and the location within that page. Mapping tables, maintained by the supervisor, allow symbolic addresses assigned at compile-time to be mapped into absolute, paged addresses at run-time. When the task executes, pages of program code are swapped between core and disk with a high degree of autonomy from the main processor thereby enabling another task to be executed while the processor waits on the remainder of the program it was executing to be loaded **(Figure 4).** ■

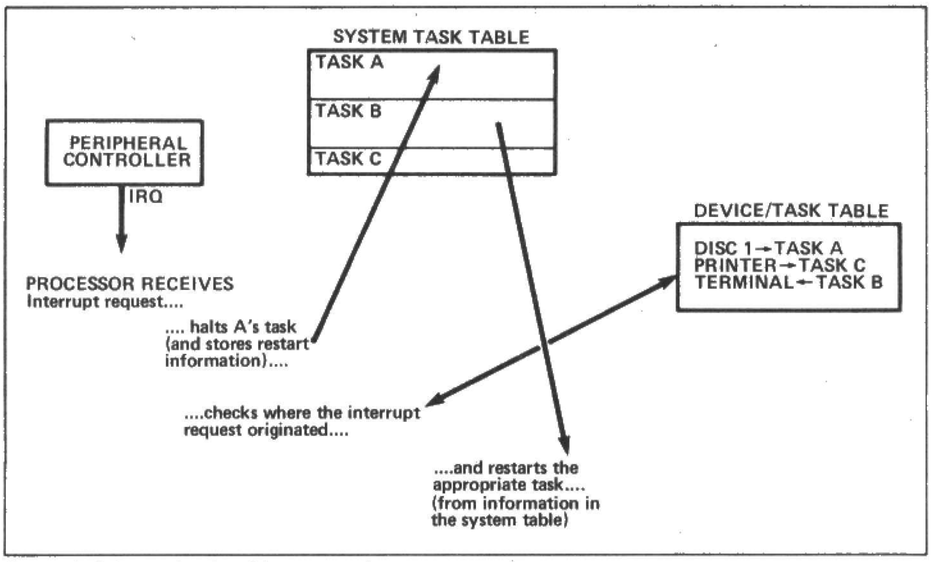
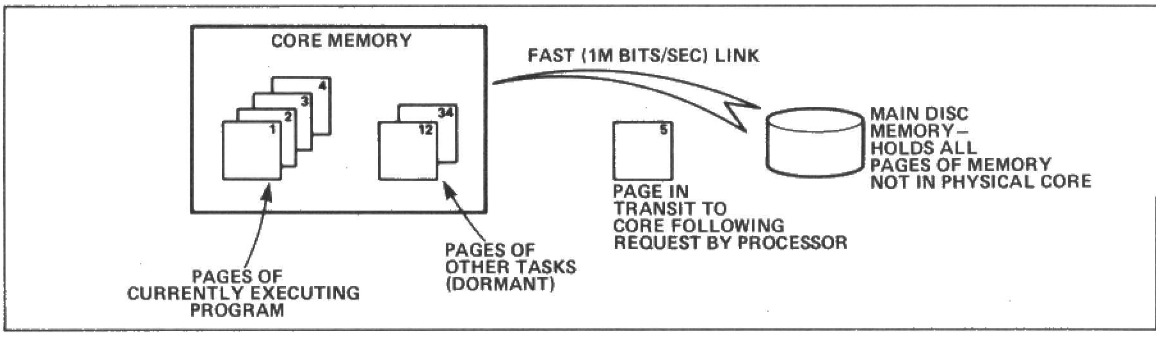

SYSTEM TASK TABLE
TASK A
TASK B
TASK C
PERIPHERAL CONTROLLER
IRQ
PROCESSOR RECEIVES
Interrupt request....
.... halts A's task (and stores restart information)....
....checks where the interrupt request originated....
....and restarts the appropriate task.... (from information in the system table)
DEVICE/TASK TABLE
DISC 1→TASK A
PRINTER→TASK C
TERMINAL←TASK B

Figure 3. Schematic of multiprogramming system.


CORE MEMORY
1
2
3
4
12
34
5
FAST (1M BITS/SEC) LINK
MAIN DISC MEMORY– HOLDS ALL PAGES OF MEMORY NOT IN PHYSICAL CORE
PAGE IN TRANSIT TO CORE FOLLOWING REQUEST BY PROCESSOR
PAGES OF CURRENTLY EXECUTING PROGRAM
PAGES OF OTHER TASKS (DORMANT)

Next Month: Multiprocessing

Figure 4. Simple virtual memory system.

hobbyboard

The Complete Printed Circuit Workshop

NEW FULL HOBBYBOARD CATALOGUE & PRICE LIST

Supplies all your PCB needs including Draughting Aids, Process Materials, Hand Tools, Hardware, Instrument Cases etc. Also introducing a choice of PC Prototyping Systems and the Daylight Fototool System for precision photography (no darkroom) and superb label/panel-making. PLUS 2 pages of Add-ons which give your home computer useful "real world" applications and offer fascinating program/hardware development.

Newly Published Full Catalogue - price £1.50
(Refundable with 1st order over £10)

Easy Add-ons for ZX Spectrum & ACE

NEW

17 exciting electronic projects to build and run your own micro.

- LIGHT PEN
- PICTURE DIGITISER
- KEY PAD
- MODEL CONTROLLER
- WEATHER STATION
- + OTHER EXCITING & INTERESTING PROJECTS

A newly released book by well known author Owen Bishop, published by Bernard Babani, describes in detail 17 projects to build (using low cost materials) to bring your micro into the real world.

The most complex component (the DECODER) is supplied in kit form ready to assemble with all components and plated through PCB

Order ref HB/2000 "EASY ADD-ONS" BOOK + DECODER KIT	£24.00
Order ref HB/2001 "EASY ADD-ONS" BOOK ONLY	£3.00
Order ref HB/2002 DECODER KIT ONLY	£22.00
Order ref HB/2003 DECODER PCB ONLY	£8.00

Please state computer when ordering. Prices inclusive of VAT & carriage

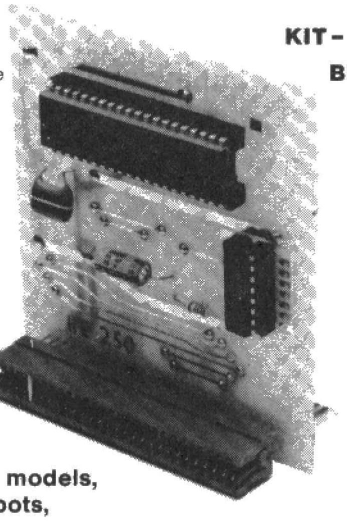
hobbyboard
complete P.C.B. workshop

a division of
KELAN ENGINEERING Ltd
Hookstone Park
Harrogate, N. Yorks

ECM/2

INPUT/OUTPUT PORT FOR ZX SPECTRUM

- 3 Channels (24 lines)
- Programmable mix of input and output
- May be used with printer
- Uses IN/OUT instructions
- Up to 8 may be connected



KIT – £16.95
BUILT – £18.95
(Output connector £3.25 if required)

Includes VAT and Inland Postage

Ideal for controlling models, motors, robots, automatic equipment, security systems etc. practical applications circuits included.

Export postage (surface) £2.50. Send SAE for full catalogue of Spectrum and ZX81 accessories. Cash with order or ACCESS.

ECM02

REDDITCH ELECTRONICS
21 FERNEY HILL AVENUE,
REDDITCH, WORCS B97 4RU

SOFTWARE REVIEWS

A selection of utility software

Editor Assembler

(Spectrum 16 and 48K)
Picturesque

The Spectrum computer needs all the help it can get when it comes to increasing its speed, and one of the most popular methods involves the use of machine code. This is an extremely tedious language to use as it stands, consisting merely of numbers, so some bright spark invented assembly language, which uses **mnemonics** (a memory aid) for each of these different numbers. This makes life a lot simpler and machine code programming considerably less arduous.

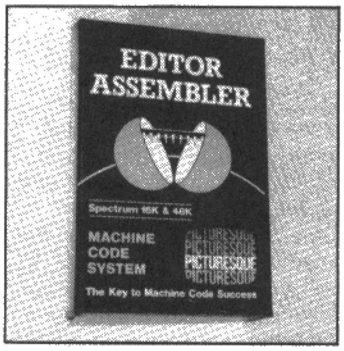
The problem is that a special program is required to convert assembly language (that the human understands) into machine code (that the computer understands). Such a program is called an assembler, and Picturesque have produced one for the Spectrum. Called simply **Editor Assembler,** it comes in a box the size of the standard Sinclair boxed software, along with a 30 page booklet euphamistically described as a manual. Those that have never used an assembler before may be puzzled by the 'Editor' in the title; it means that the package comes with a system to alter and delete assembly language that the programmer is currently developing.

Although the program is suitable for both 16K and 48K machines, it obviously makes a lot more sense to use a 48K Spectrum; not only for the 32K extra memory obtained this way, but also because a programmer can develop software with higher potential usage.

Loading the cassette is no problem, as it has a BASIC front end loader, and simply typing LOAD"" will load and run the program. As an assembler is of little use to anyone without any assembly text to actually assemble, running the program actually invokes the editor first. This enables the user to enter text, using a line numbering system similar to a BASIC program; edit the text a line at a time (this is why this sort of editor is called a line editor), with the ability to renumber the lines; list the program so far; delete lines, and all the usual things you expect from a line editor. It does have its limitations, though. First of all, line editors by their very nature are fairly cumbersome to use, more accomplished programmers preferring what is known as a screen editor, in which there are no line numbers, and in fact the whole thing is much closer to a sophisticated word processor than anything else. Unfortunately, such a system involves heavy memory usage, and there are no usable screen editors at all on the Spectrum to my knowledge. Picturesque have made things a little better by writing software within the editor that gives a 40 column screen rather than the normal Spectrum 32 column variety.

However, other limitations involve a rather minimal character set, and to obtain the more obscure characters one has to read the manual fairly closely. Text is always displayed as white lettering on a blue screen, which, fortunately, is a reasonable combination for it cannot be altered. Particular editor commands that this version supports are AUTO, which prompts with a line number for each new line, LIST, EDIT, which allows editing of a specified line, NEW, which removes all of the current text and resets all the editor pointers to their original values, RETURN, which takes the user back to BASIC, and RENUM, which renumbers the lines present with the increment specified.

As with all utilities that do their jobs, this editor is fine and quite suitable for the application, but when compared to some others, its deficiences begin to become apparent. The Hisoft Devpac line editor is far more comprehensive than this one, and as that product is itself an editor/assembler, further comparisons will be made later.



Having entered and edited your text, the next step is to assemble it into the 'object code' — the machine code. To this end, a number of standard assembler 'directives' are supported, and these tell the assembler where to put the code, give values to labels, fill particular memory locations with numbers that don't necessarily correspond to any machine instructions, etc. Here is a brief summary:

Semi colon: this indicates that the rest of the text on this line is a comment (as in BASIC's REM statement), and is to be ignored.

EQU: this means associate the label preceeding this directive with the value following it. Once a label has been defined this way, it cannot be redefined later in the program.

DEFL: is basically the same as EQU, except that the label can be redefined. This is actually bad programming practice and is not to be recommended.

DEFB,DEFW,DEFM: mean put numbers into the relevant memory locations. DEFB means a byte at a time, DEFW means a 2 byte word at a time, and DEFM means the ASCII string following the DEFM.

DEFS: this simply reserves the specified number of bytes within the program.

ORG: this tells the assembler where to write the code to.

In use, the assembler does exactly what it is supposed to do, but in common with nearly all assemblers for the Spectrum, the Picturesque assembler has forgotten its prime function: as a tool. As such, it should cause as little a problem as possible to its user, and in this respect, just like almost all the others, it fails. The limitation of having to have the assembler itself in a particular and unalterable part of memory is ridiculous, as is the predefined locations for text buffers, object buffers, and label tables. Experience shows that to write a 7K program in assembler requires typically 40K of text, and this assembler simply cannot handle this in one go. Compare Hisoft's Devpac, which has far more useful attributes: it is totally relocatable to anywhere in memory, it can assemble text from tape or microdrive rather than having to have it in memory, and it has such luxuries as conditional assembly, a full printer interface, unbeatable speed and is far far cheaper than Picturesque's offering. Look before you buy.

Spectrum Monitor

(Spectrum 16 and 48K)
Picturesque

To accompany the assembler described above, Picturesque also supply a machine code monitor in the same sort of packaging. A monitor is a utility that enables the programmer to examine in detail the operation and effects of his machine code program, looking at the registers and memory locations involved. This is just what this package does, but it suffers from the same problems as the assembler. It has to be loaded to one predefined specific address in memory, which means that any user program in this area is obliterated — very useful.

What this program does — and all monitors must — is display and alter memory locations, copy and contents of RAM from one location to another, set breakpoints in programs, display and alter the registers, and disassemble sections of memory. BUT it does not allow the user to step through his programs one instruction at a time, and only one breakpoint at a time can be set. Now, as we mentioned in the assembler review above, a company called Hisoft market a product called Devpac which includes a monitor with none of these limitations. Not only is it totally relocatable, but it can actually disassemble memory and produce a text file for its accompanying assembler.

A monitor is a very useful tool, but it must be far more comprehensive than this one to be considered such.

Abersoft FORTH

(Spectrum 16 and 48K)
Melbourne House £14.95

Approximately six months ago, a small Welsh Company called Abersoft released the best FORTH program available for the Spectrum. Since then, the much larger Melbourne House software house has bought all rights to this program, and have re-packaged and re-released it under the inspired name of **Abersoft FORTH.** It still costs £14.95, and the booklet supplied with it is virtually unchanged, but of course the program itself still excels. Since the original release of this product, Sinclair Research themselves have adopted and released Artic FORTH, at the same price.

We recently had the dubious pleasure of comparatively testing these packages, and there was not one aspect in which the Sinclair package got even close to Abersoft's.

As discussed in last month's issue, FORTH is a fairly versatile and flexible language, touted as being an easier machine code, in that it almost has a semantic structure, even if its syntax leaves a lot to be desired, and it is very fast. In fact, the above mentioned tests showed that Abersoft FORTH is approximately four times the speed of Sinclair's in the majority of applications. Considering that even Sinclair FORTH is some four or five times faster than Spectrum BASIC, the improvement offered by Abersoft FORTH can easily be seen.

Abersoft FORTH is based on the most popular standard for the language, Fig-FORTH, and is in fact a little more comprehensive than most, offering even more facilities and prefefined words than most other versions.

There are an awful lot of people about who have been convinced by the FORTH fables, and Melbourne House have been very astute in adopting this excellent version of the language.

BOOK REVIEWS

Harry Fairhead's monthly guide

The Friendly Computer Book

Jonathan Inglis
BBC Publications

This is a careful and comprehensive introductory guide to computer programming using the three machines that have been widely purchased by schools as a result of the Department of Industry's microcomputers in schools scheme, namely the BBC Micro, the Sinclair Spectrum and the Research Machines 380Z. The author has approached the difficult task of dealing with three distinctly different machines in a single volume with commendable thoroughness although the fact that much of the material has to be presented separately for each of them contributes to the impression that this is an encyclopaedic work. Obviously, with so much ground to cover, the book cannot go extensively into the special or more advanced features of any of the three computers.

Accordingly, there are just outline details of the BBC Micro's and the Spectrum's graphics and sound commands and only the graphics facilities standard on the unexpanded 380Z are discussed. The introduction provided should give the user confidence to tackle his computer's own manual.

As far as I'm concerned this book scores points for presentation. It is spiral bound and so stays conveniently open at the correct page while you experiment at the keyboard. I was immediately attracted by the cover and enjoyed the cartoon characters who appear throughout the book. They helped both to convey ideas and to provide light relief — this really is a user-friendly book. I can recommend this book for school teachers and for family use — especially if it's the book used in your child's classroom. The drawback it inevitably has for the home is that you are likely only to have one of the three computers available — it's a pity that the BBC did not ask the author to prepare three slimmer volumes rather than one, all-embracing one.

Electronic Speech Synthesis

Geoff Bristow (Ed)
Granada

One of my current problems is finding technical literature that really gives me the sort of advanced information that I want. The trouble is that either the level is too low and practical or too high and theoretical! Electronic Speech Synthesis is a collection of papers each one written by an expert in the topic. Getting such a large number of people to produce one book sounds like a recipe for disaster but it has resulted in a surprisingly good and valuable book! The trick seems to be that each one of the contributions was sought out to fit into an overall framework that the editor, Geoff Bristow, had formulated. This preplanning has resulted in a book that is very different from the usual random collections of papers that eminate from conferences etc. Instead the book is logical and forms a good introduction to the theory and techniques of speech synthesis.

Divided into three parts, techniques, technology and applications, the book begins with an examination of the principles that lie behind human speech production. The middle section concentrates on examples of chips that use the various techniques of time domain synthesis, LPC and phonetic synthesis. The final section is about applications and often reads like a catalogue of what could be done. It is still lively and interesting however, and should stimulate one or two of its readers actually to do something!

This is a book that deals with an advanced subject in an advanced way and it doesn't avoid including mathematics where necessary but neither is it ashamed of including circuit diagrams. My final conclusion is that it is not a book for the beginner but it is essential reading for any aspiring expert in the field of speech synthesis, signal processing or (to a lesser extent) AI. As befits a books that you will want to keep around for reference over a long period it is in hardback and is attractively produced with clear diagrams throughout.

LETTERS TO THE EDITOR

Send your letters to The Editor, E&CM, 155 Scriptor Court, Farringdon Road, London EC1R 3AD

Sir,

I am a regular reader of your magazine and was particularly interested in the article concerning the Brother EP22 printer/typewriter, on which this letter was typed.

I have a 48K Spectrum with the Fuller RS232C interface and would assure your readers that there is total compatibility with this set-up.

The interface provides the software for LLIST and LPRINT and offers both 75 and 300 baud rates. It is simple to write a COPY routine in BASIC, by using SCREEN$ in a loop scanning the 32 characters for each of the 22 lines. Of course, the Brother EP22 is limited to its own character set, so having graphics on the screen produces interesting results.

I am writing a word processor program specifically for the EP22 with my interface configuration and as an 'end user' would highly recommend this to other Spectrum owners.

A. Katz
Edgware, Middlesex.

Sir,

I have been following your series on the 68705 chip with great interest and have been awaiting the January issue for the assembler so that I can start experimenting.

Now I have the January issue I feel somewhat let down. The whole article assumes that everyone is familiar with 68705 opcodes and their mnemonics, although as a 6502 user most of them are obvious, indeed 80% of them are identical to 6502, I feel that some of the Z80 brigade feel hopelessly lost.

Surely some explanation of the addressing modes and test/branch codes peculiar to this chip could have been included somewhere in the series.

I hope that you might find space in a future issue to correct this omission.

D. J. Cramer
Crawley, W. Sussex.

EDITOR'S NOTE
While we included as much information as we could on 68705 opcodes in the cross-assembler article, your point about addressing modes and test/branch codes is noted, and we will be returning to this subject in a future article. In the meantime however, anyone who would like further details of the 68705 should refer to the Motorola handbook.

Sir,

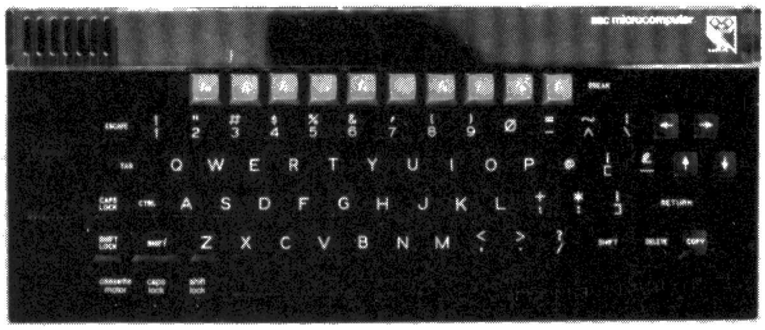
I wish to gain a sound, practical understanding of the fundamental principles of electronics and electronic circuitry, and in particular, their application to computers, logical and digital circuits. To this end, I bought the December issue of your magazine in the hope of gaining a lead as to available home study courses and construction kits providing the necessary combination of theoretical and practical knowledge, beginning at the simplest, most elementary level and progressing systematically towards the more advanced and complex aspects.

But the welter of unfamiliar jargon was too much for me, completely incomprehensible to me I'm afraid. I wonder, therefore, whether you and your knowledgably expert staff may be able to help me by kindly pointing me in the right direction with the names and addresses of those supplying appropriate instruction courses and construction kits.

C. S. Brindley
Lymington, Hants.

EDITOR'S NOTE
Well the first direction to point you in is the free booklet on the front cover of this issue, 'Exploding the myths of electronics and computing' which has been designed specifically for people with little knowledge of digital electronics. The second line of inquiry should be to the Technician Education Council (TEC) who administer a number of home study courses on electronics, and have produced an excellent series of books on microelectronics and computing for various levels of competence. The address is *TEC, Woburn Place, London W2.*

Experimental science with the BBC micro



Last month David Knaggs described how to set up the BBC micro to measure and collate data in a scientific experiment — the measurement of the coefficient of resistance. In part 2 the software and the results are described.

Using the BBC micro's analogue to digital converter a standard physics experiment was performed to measure the temperature coefficient of resistance of different materials.

The selection of suitable transducers, together with the inclusion of the relevant calibration data in the driver software, allow the BBC machine to give digital representations of quantities such as voltage, current, resistance, temperature, force etc. directly from the experiment. The scheme described in part 1 allowed sufficient data to be collected to give values for the temperature coefficient of resistance of samples of Copper, Manganin and for a Thermistor, which may be expected to show a variation from a positive value, through an almost zero value to a negative value. The results of the experiment are given below. The system was found to produce consistent results over a large number of determinations, typical examples are shown opposite. Average values for each material are as follows:

Copper	Alpha	= 0.004 per K
Manganin	„	= 0.0005 per K
Thermistor	„	= −0.016 per K

These values are in good agreement with those given in the standard tables and would seem to suggest that the method is reliable. The values for Manganin show most variation but the changes in resistance which were observed seemed to suggest that these variations were a property of the material rather than an effect due to the method used. The values given for the resistance at zero Centigrade were in good agreement with those obtained prior to the experiment at room temperature. The graphical representation of the results is shown in **Figure 1.**

> "... effective use of the microcomputer as a high speed calculator and data gatherer"

Program Details

A complete listing of the program is given in **Listing 1.** The following notes are intended to assist in following the operation of the program.

a) Variable Assignments

N	— experiment cycle counter
S(n)	— value of series resistor with n identifying the specimen
F	— oscillator 'frequency'
Y,Z	— contents of the low and high order counter registers after counting period
X	— differences between consecutive 'frequency' determinations
V(n,N)	— measured voltages
I(n,N)	— calculated currents
R(n,N)	— specimen resistances
T(N)	— temperature calculated for each cycle
G	— gradient of resistance - temperature graph
RI(n)	— resistance at zero C
A(n)	— temperature coefficient of resistance

b) Procedure Definitions

PROCtemp - measures the 'frequency' of the oscillator and converts this value to a temperature in degrees Centigrade.

PROCvolt - records the output of each channel of the ADC and converts these to voltages.

PROCcalc - calculates the current flowing in each branch of the circuit and uses this value to give the resistance of each component.

PROCdisplay - gives a visual report on the operation of the experiment and displays the current values of temperature, voltage, current and resistance for each specimen.

PROCresults - presents the full results for each specimen along with the value of the resistance at zero Centigrade and the temperature coefficient of resistance.

PROCmenu - gives the user the range of output options which are available from the program.

Conclusion

The method described here has some limitations but in the classroom situation these would not prove too serious. The first of these involves the method used to measure the temperature of the oil bath. The graphical representation of the calibration data, with the approximation of the resulting curve to straight line segments of differing gradient, is likely to introduce a small error into the system. This error was estimated to be approximately 1.3% in the prototype system. During the calculation stage a second source of error is introduced, namely in the calculation of the resistance at zero Centigrade. This was found simply from the gradient of a line which was imagined to join the first and last measured points, on the assumption that the relationship between the resistance and temperature was linear. The graphs produced from the collected data (Figure 1) would seem to suggest that as far as the Copper and the Thermistor were concerned this assumption was not too in-

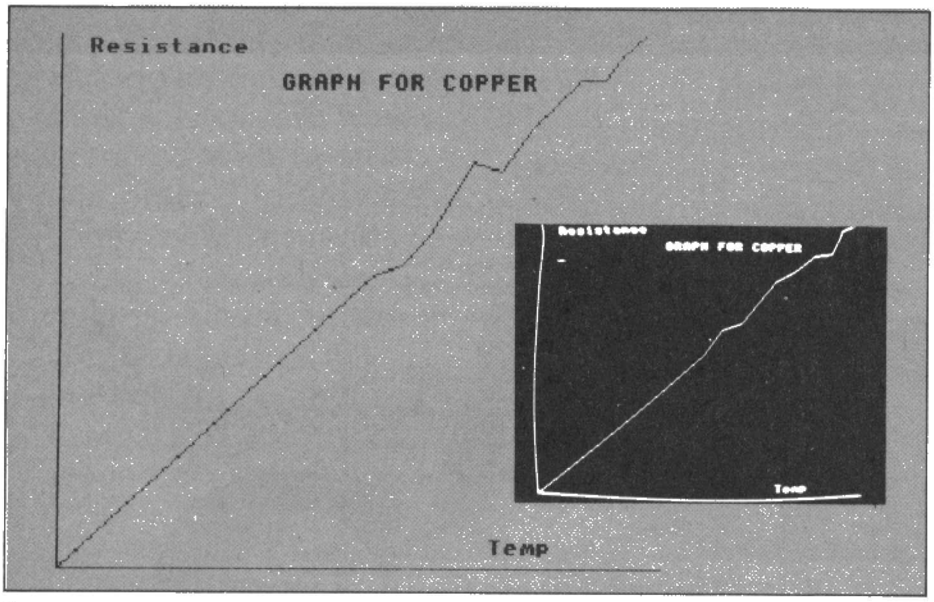


Figure 1(a)

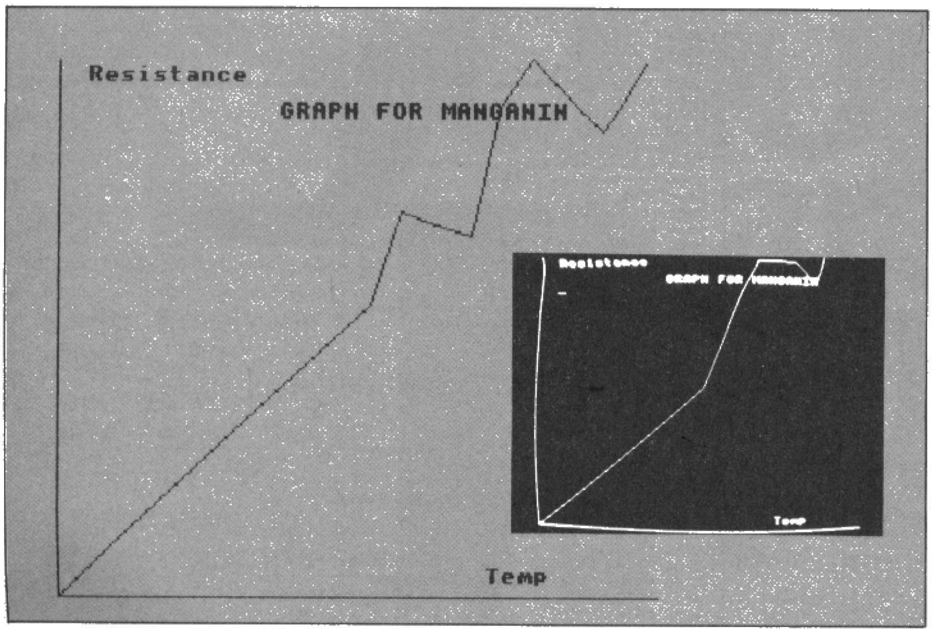


Figure 1(b)

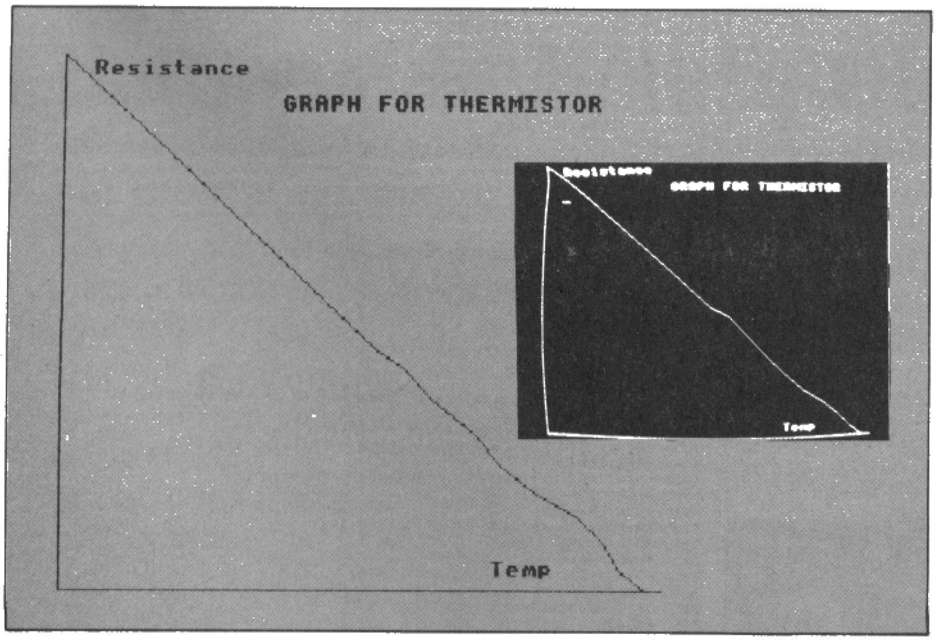


Figure 1(c)

accurate, but, as has already been mentioned, the results for Manganin were inclined to be rather erratic and frequently produced resistances which were outside the range defined by the first and last values. This variation was not allowed for and it may be worth considering the introduction of some statistical method for coping with results of this type.

The system described here is simple to set up and use and could be adapted to other standard experiments in Science without too much trouble. One advantage of such a method is that results can be obtained automatically while the relevant theory is being taught, so that differences between matérials can be illustrated quickly as and when they are required. A method of this type could save a great deal of valuable class time by quickly supplying results in cases where a full class practical is neither warranted nor necessary. ■

RESULTS FOR COPPER

Voltage	Current	Resist	Temp
0.8057	0.1544	5.2181	21.9140
0.8079	0.1543	5.2355	23.7860
0.8141	0.1538	5.2946	25.6150
0.8145	0.1534	5.3104	27.0940
0.8176	0.1525	5.3599	28.4200
0.8198	0.1523	5.3809	30.0520
0.8220	0.1525	5.3887	31.2760
0.8268	0.1518	5.4471	33.1120
0.8281	0.1521	5.4458	34.3860
0.8316	0.1516	5.4857	35.9160

Resistance at zero deg. C is 4.7992.

Temperature Coefficient of Resistance is 0.0040 per K

RESULTS FOR MANGANIN

Voltage	Current	Resist	Temp
0.7829	0.1593	4.9153	21.9140
0.7837	0.1595	4.9150	23.7860
0.7873	0.1595	4.9371	25.6150
0.7846	0.1597	4.9119	27.0940
0.7851	0.1595	4.9233	28.4200
0.7855	0.1596	4.9203	30.0520
0.7873	0.1599	4.9226	31.2760
0.7895	0.1597	4.9422	33.1120
0.7886	0.1605	4.9136	34.3860
0.7899	0.1605	4.9219	35.9160

Resistance at zero deg. C is 4.9050

Temperature Coefficient of Resistance is 0.0001 per K

RESULTS FOR THERMISTOR

Voltage	Current	Resist	Temp
0.8949	0.0019	464.0055	21.9140
0.8840	0.0020	449.3094	23.7860
0.8668	0.0020	427.0079	25.6150
0.8466	0.0021	405.6031	27.0940
0.8281	0.0021	386.8824	28.4200
0.8119	0.0022	370.0729	30.0520
0.7934	0.0023	351.2087	31.2760
0.7811	0.0023	339.5541	33.1120
0.7635	0.0024	323.2995	34.3860
0.7464	0.0024	308.7273	35.9160

Resistance at zero deg. C is 707.0256

Temperature Coefficient of Resistance is −0.0157 per K

Software ▶

LISTING 1: Temperature coefficient of resistance.

```
1020DIM V(4,10),I(4,10),R(4,10),T(10),R1(4),A(4),S(4),N$(4)
1030 MODE 4
1040S(2)= 4.7 :S(3)=4.7 :S(4)=330
1050REM SELECT LINES ON USER PORT
1060?&FE62=1 :REM ONLY PB0 IS OUTPUT
1070 N=1
1080 REPEAT
1090  PROCtemp:REM mesure temperature
1100  PROCvolt
1110   REM switch heater on :?&FE60=1
1120 ?&FE60=1
1130 TI=TIME
1140  PROCcalc
1150  PROCdisplay
1160  IF TIME<TI+5000 THEN 1160
1170   REM switch heater off:?&FE60=0
1180 ?&FE60=0
1190 DELAY=TIME
1200 IF TIME <DELAY+1000 THEN 1200
1210 N=N+1
1220 UNTIL N>10
1230PROCresults
1240 PROCmenu
1250PRINT"Do you want another set of results ?"
1260 PRINT"Type Y or N."
1270IF GET$="Y" THEN 1070
1280CLS:CLG
1290PRINT "Hope that was satisfactory. Goodbye."
1300 END
1305 REM ******************************
1310DEF PROCtemp
1320F1=0
1330?&FE6B=&20
1340REPEAT
1350  ?&FE68=&FF
1360  ?&FE69=&FF
1370 START=TIME
1380 IF TIME<START+25 THEN 1380
1390   Y=?&FE68
1400   Z=?&FE69
1410  F=65535-Y-256*Z
1420  X=F-F1
1430 F1 =F
1440 UNTIL X<5
1450?&FE6B=0
1460IF F<373 THEN T(N)=24.8-(INT(((373-F)/12.82)*1000))/1000
1470IF F>373 THEN T(N)=24.8+(INT(((F-373)/19.61)*1000))/1000
1480 ENDPROC
1485 REM******************************
1490DEF PROCvolt
1500 FOR J=1 TO 4
1510 V(J,N)=(ADVAL(J)/16)*(1.8/4095)
1520NEXT J
1530ENDPROC
1535 REM******************************
1540DEF PROCcalc
1550FOR J=2 TO 4
1560 I(J,N)=(V(1,N)-V(J,N))/S(J)
1570 R(J,N)=V(J,N)/I(J,N)
1580NEXT J
1590ENDPROC
1595 REM******************************
1600DEF PROCdisplay
1610CLS
1620PRINT "Cycle No.=   ";N:PRINT:PRINT
1630 PRINT "Most recent results are :- "
1640 PRINT
1650 PRINT"Temperature =   ";T(N);"deg C"
1660 PRINT
1670PRINT"Supply voltage =   ";V(1,N)
1680 PRINT
1690 PRINT"Voltage across ";
1700  FOR J=2 TO 4
1710 PRINTTAB(14);"component ";J;"= ";V(J,N)
1720 NEXT J
1730 PRINT
1740 PRINT"Resistance of ";
1750 FOR X=2 TO 4
1760 PRINTTAB(14);"component ";X;"= ";R(X,N)
1770 NEXT X
1780ENDPROC
1785 REM******************************
1790DEF PROCresults
1800FOR J=2 TO 4
1810 CLS
1820  IF J=2 PRINT"Results for COPPER"
1830  IF J=3 PRINT"Results for MANGANIN"
1840  IF J=4 PRINT"Reslts for THERMISTOR"
1850  PRINT:PRINT
1860PRINT"Voltage","Current","Resist","Temp"
1870    FOR N=1 TO 10
1880 @%=&2030A
1890    PRINT ;V(J,N),;I(J,N),;R(J,N),;T(N)
1900    NEXT N
1910  G=(R(J,10)-R(J,1))/(T(10)-T(1))
1920  R1(J)=R(J,1)-(G*T(1))
1930 PRINT:PRINT
1940  PRINT "Resistance at zero degC is  ";R1(J)
1950 A(J)=(R(J,10)-R1(J))/(R1(J)*T(10))
1960 PRINT:PRINT
1970  PRINT"Temperature Coefficient of Resistance is ";A(J);"pe
1980  TIME =START
1990  IF TIME<START+1000 THEN 1990
2000 @%=10
2010NEXT J
2020 CLS
2030ENDPROC
2035 REM******************************
2040DEFPROCgraph
2050 FOR J= 2 TO 4
2060 CLS :CLG
2070VDU 29,40;40;
2080MOVE 0,0:DRAW 1200,0
2090MOVE 0,0:DRAW 0,960
2100MOVE 0,0
2110VDU 28,28,31,36,29
2120VDU 31,30,30
2130VDU 4:PRINT"Temp"
2140VDU 28,3,4,36,1
2150 N$(2)="COPPER"
2160 N$(3)="MANGANIN"
2170 N$(4)="THERMISTOR"
2180VDU 31,40,30
2190 VDU 4:PRINT"Resistance "
2200 VDU 31,12,2
2210 PRINT"GRAPH FOR ";N$(J)
2220 IF (R(J,10)-R(J,1))>0 THEN PROCpos ELSE PROCneg
2230 TIME=START
2240  IF TIME <START+2500 THEN 2240
2250 NEXT J
2260VDU 26
2270ENDPROC
2275 REM******************************
2280DEF PROCprint
2290PRINT"Is your printer a Serial or a Parallel type?"
2300 PRINT"Type S or P."
2310 IF GET$="S" THEN *FX5,2 ELSE *FX5,1
2320 *FX 6,0
2330PRINT"How many columns has your printer?"
2340INPUT C
2350WIDTH C
2360VDU 2
2370FOR J=2 TO 4
2380  IF J=2 THEN PRINT"Results for COPPER"
2390  IF J=3 THEN PRINT"Results for MANGANIN"
2400  IF J=4 THEN PRINT"Results for THERMISTOR"
2410  PRINT"Voltage","Current","Resist","Temp"
2420 @%=&2030A
2430    FOR N=1 TO 10
2440 PRINT ;V(J,N),;I(J,N),;R(J,N),;T(N)
2450    NEXT N
2460PRINT:PRINT
2470PRINT"Resistance  zero deg C is   ";R1(J)
2480PRINT:PRINT"Temperature Coefficient of Resistance is ";A(J);"per K"
2490 @%=10
2500NEXT J
2510 VDU 3
2520 ENDPROC
2525 REM******************************
2530 DEF PROCpos
2540 MOVE 0,0
2550 FOR N=1 TO 10
2560 X=(T(N)/T(10))*1200
2570 Y=(R(J,N)-R1(J))*(960/(R(J,10)-R1(J)))
2580 DRAW X-30,Y
2590 NEXT N
2600 ENDPROC
2605 REM******************************
2610 DEF PROCneg
2620 MOVE 0,960
2630 FOR N= 1 TO 10
2640 X=(T(N)/T(10))*1200
2650 Y=(R(4,N)-R(4,10))*(960/(R1(4)-R(4,10)))
2660 DRAW X-30,Y
2670 NEXT N
2680 ENDPROC
2685 REM******************************
2690 DEF PROCmenu
2700 PRINT"These are the output"
2710 PRINT"options which are available."
2720 PRINT :PRINT
2730 PRINT"1. GRAPH"
2740 PRINT:PRINT"2. PRINTOUT"
2750 PRINT:PRINT"3. SAVE DATA TO TAPE"
2760 PRINT:PRINT"4. RESULTS SUMMARY"
2765 PRINT:PRINT"5. REPEAT OF RESULTS DISPLAY"
2770 PRINT:PRINT
2780 PRINT"Type the number required"
2790 A$=INKEY$(800)
2800 IF A$="1" THEN PROCgraph
2810 IF A$="2" THEN PROCprint
2820 IF A$="3" THEN PROCcas
2830 IF A$="4" THEN PROCsum
2835 IF A$="5" THEN PROCresults
2840 CLS:PRINT"Type Q to quit"
2850 PRINT"Type R to return to the menu"
2860 PRINT:PRINT
2870  IF GET$="Q" THEN 1250 ELSE 2700
2880 ENDPROC
2885 REM******************************
2890 DEF PROCcas
2900 PRINT:RRINT"Position the tape correctly to receive data"
2910 S=OPENOUT("RESULTS")
2920 FOR J=2 TO 4
2930 FOR N=1 TO 10
2940 PRINT£S, V(J,N),I(J,N),R(J,N),T(N)
2950 NEXT N
2960 PRINT£S,R1(J),A(J)
2970 NEXT J
2980 CLOSE £S
2990 ENDPROC
2995 REM******************************
3000 DEF PROCsum
3010 CLS
3020 PRINT"This is a summary of the results of the experiment."
3030 PRINT:PRINT:
3040 PRINT"Starting temperature = ";T(1);" deg C"
3050 PRINT:PRINT"Maximum temperature = "T(10);" deg C"
3060 PRINT:PRINT
3070 PRINTTAB(10);"COPPER","MANGANIN","THERMISTOR"
3080 @%=&2040A
3090 PRINT"Resist."
3100 PRINT"at zero",;R1(2),;R1(3),;R1(4)
3110 PRINT:PRINT"Temp"
3120 PRINT"coeff of"
3130 PRINT"resist.",;A(2),;A(3),;A(4)
3140 START=TIME
3150 IF TIME<START+500 THEN 3150
3160 @%=10
3170 ENDPROC
```

Spectrum speech synthesiser

In Part 2 of Robert Harvey's project to build a £15 speech synthesiser, the software and PCB details are given.

To recap, the speech synthesiser is based on the General Instrumenmt SPO256 chip. While the board is designed with the Spectrum specifically in mind, it can easily be interfaced to any Z80 based micro, and as the software is written in BASIC, this too should be transportable with little modification.

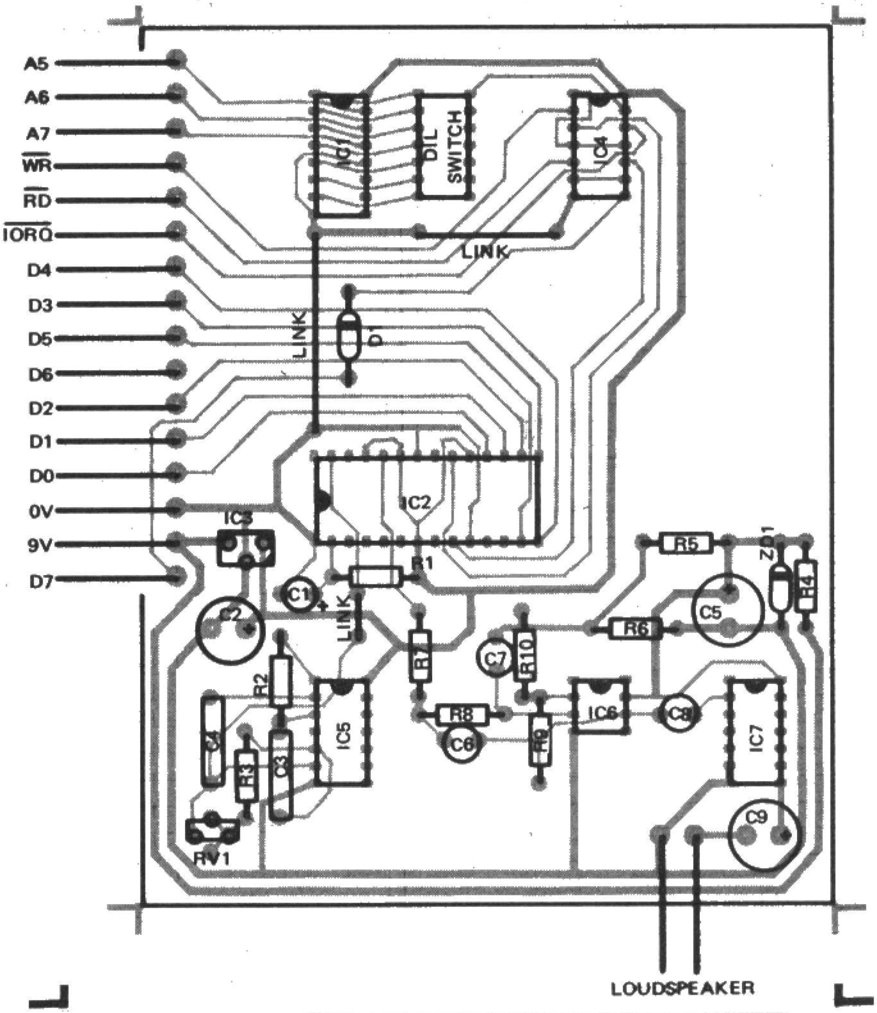
The circuit is easily constructed on Vero VQ Board using a wiring pen, or by using a PCB. The dual 28-way connector can be soldered directly to the board and thus the finished unit (with speaker mounted on the board) can plug in and stand up behind the Spectrum. The circuit includes a 5V power supply for the logic and it is recommended that this be used because the SPO256 alone uses around 90mA, and it is well known that the Spectrum's own internal power supply will not stand much more loading. The software is shown below in Listing 1. ■

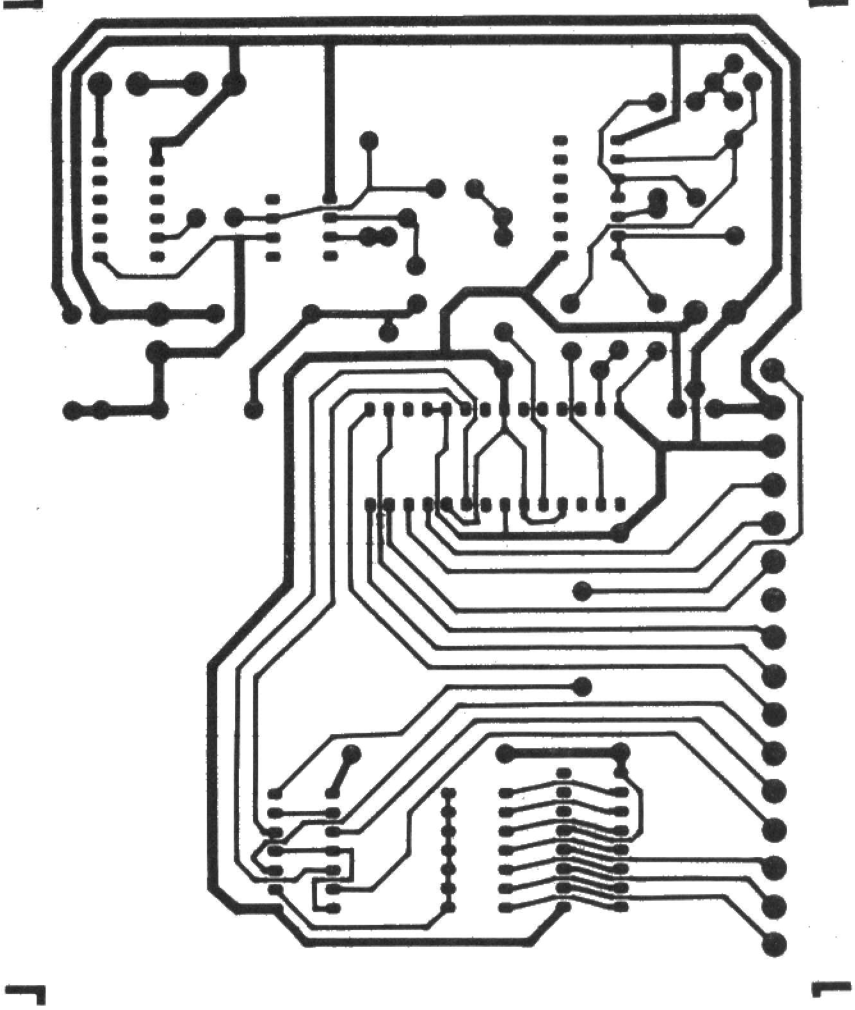
LISTING 1

```
10   PRINT "I AM A ZX SPECTRUM COMPUTER"
20   LET X=300: GO SUB 1000: REM *** NOW SAY IT
30   PAUSE 50
40   PRINT "PLEASE ENTER A NUMBER"
50   LET X=310: GO SUB 1000
60   LET A$=INKEY$: IF A$<"0" OR A$>"9" THEN GO TO 60
70   LET A=VAL A$
80   PRINT "THE NUMBER IS ";A
90   LET X=320: GO SUB 1000
100  LET X=A+200: GO SUB 1000
110  PAUSE 25
120  GOTO 40


199  REM **** NUMBERS ONE TO
     NINE IN ALLOPHONES
200  DATA 43,12,12,39,53,0: REM ZERO
201  DATA 46,23,11,0: REM ONE
202  DATA 13,31,0: REM TWO
203  DATA 40,14,19,0: REM THREE
204  DATA 40,58,0: REM FOUR
205  DATA 40,6,40,0: REM FIVE
206  DATA 55,12,2,41,55,0: REM SIX
207  DATA 55,7,35,7,11,0: REM SEVEN
208  DATA 20,2,13,0: REM EIGHT
209  DATA 56,6,11,0: REM NINE


299  REM **** SENTENCES IN ALLOPHONES (See PRINTs)
300  DATA 6,3,26,16,3,20,3,43,7,7,21,4,7,1,41,55,3,55
301  DATA 9,7,41,17,14,15,16,3,42,15,16,9,49,22,13,51,0
310  DATA 9,45,19,43,3,7,11,2,13,51,3,20,3,11,15,16,1
311  DATA 28,51,0
320  DATA 18,52,3,11,15,16,1,28,51,3,12,12,55,4,0
999  REM **** SUBROUTINE TO SPEAK A PHRASE
1000 RESTORE X
1010 READ IN
1020 IF IN 159>127 THEN GO TO 1020
1030 OUT 159,N
1040 IF N THEN GO TO 1020
1050 RETURN
```





"THERE MUST BE A COMPUTER DEALER I CAN TURN TO FOR GUIDANCE...."

If you're still staggering through the computer jungle and not getting sensible answers to your questions, we have some good news:

Now you can turn to **professional** people who are capable of giving you sound advice on practically all aspects of popular computing.

They all have one thing in common:

They are **COMPUTERS FOR ALL** dealers.

A Computers for All dealer is different from the normal Computer retailer.

Not surprisingly he won't try to sell you things like cameras or cosmetics; stationery or sealing wax.

He will, however, be capable of answering sensibly almost any question you have on computers and computing, and have readily available a wide range of popular computers, hardware, software, books, and peripherals.

So why not call in at your local **COMPUTERS FOR ALL** dealer today?

He can lead you in the direction you want to travel.

The shops where people matter:

AVON

MERCATOR COMPUTER SYSTEMS
3 Whiteladies Road, Clifton,
Bristol. 0272 731079.

BERKSHIRE

KENNETH WARD COMPUTERS
Verve House, London Road,
Sunningdale. 0990 25025.

CHANNEL ISLANDS

MEGA LTD.
7 Anley Street, St. Helier,
Jersey. 0534 72263.

MICROTEL
Camp-Du-Roi Vale
Guernsey, C.I.
0481 53436.

CHESHIRE

THE COMPUTER CENTRE
68 Chestergate, Macclesfield.
0625 618827

CORNWALL

COMPUTAVISION
4 Market St. St. Austell
Cornwall. 0726 5297.

FAL-SOFT COMPUTERS
8 St. George's Arcade,
Falmouth TR11 3DH. 0326
314663.

DEVON

A & D COMPUTERS
"Computerland", 6 City Arcade,
Fore Street, Exeter.
0392 77117.

BITS AND BYTES
44 Fore Street, Ilfracombe,
N. Devon EX34 9JD. 0271 62801

COMPUTER SYSTEMS (TORBAY)
Pump Street, Brixham.
08045 6565/6.

CRYSTAL COMPUTERS
209 Union Street, Torquay.
0803 22699.

SYNTAX LTD.
Midhurst, Grenofen,
Tavistock, Devon.
0822 2392.

DORSET

DENSHAM COMPUTERS
329 Ashley Road, Parkstone,
Poole. 0202 737493.

ROBERTS ELECTRICAL (BLANDFORD) LTD.
14 Market Place, Blandford
Forum.

ESSEX

AKHTER INSTRUMENTS
Unit 19, Arlinghyde Estates,
South Road, Harlow.
0279 412639.

CHELMSFORD COMPUTER CENTRE
3 Baddow Road, Chelmsford.
0245 356834.

COMPUTERAMA
88-90 London Road,
Southend-on-Sea,
Essex. 0702 335443.

COMPUTERS FOR ALL
72 North Street, Romford.
0708 752862.

HANTS

MICRO VIDEO STUDIOS LTD.
60 Normandy Street, Alton, Hants
0420 82055.

HERTS

VIDEO CITY
45-47 Fishers Green Road,
Stevenage. 0438 53808.

THE COMBINED TRADING CO.
10 & 11 Salisbury Square,
Old Hatfield. 07072 65551.

KENT

ANIROG COMPUTERS
29 West Hill, Dartford.
0322 92513.

APHROS SOFTWARE CO.
47 Hawley Square, Margate.
0843 294699.

DATA STORE
6 Chatterton Rd., Bromley, Kent.
01-460 8991.

JUTEA LTD.
92 High Street, Bridge.
Canterbury. Kent. 0227 831183.

LEAKES LTD.
63 Sidcup High Street,
Sidcup, Kent. 01-300 1142.

LANCS

P.C.S.
39 Railway Road, Darwen.
0254 776677.

4MAT COMPUTING
67 Friargate, Preston.
0772 561952.

SUMLOCK
Royal London House,
198 Deansgate,
Manchester M3 3NE.
061 834 4233.

LEICS

DIMENSION
27-29 High Street, Leicester.
0533 57479.

MOVIES COMPUTER CENTRE
5 Church Street, Melton Mowbray.
0664 61169.

LONDON

EROL COMPUTING
125 High Street, Walthamstow,
London E17. 01-520 7763.

KAYDE HOME COMPUTERS
1 Station Approach, New Eltham,
London SE9. 01-859 7505.

KELLY'S COMPUTERMARKET
227 Dartmouth Road,
Sydenham, London SE26 4QY.
01-699 4399/6202.

MICRO ANSWERS LTD.
70-71 Wilton Road, Victoria,
London SW1. 01-630 5995.

MIDDLESEX

SCREENS MICROCOMPUTERS
6 Main Avenue, Moor Park,
Northwood. 09274 20664.

TWILLSTAR COMPUTERS
17 Regina Road, Southall.
01-574 5271.

N. IRELAND

D. V. MARTIN LTD.
13 Bridge Street, Belfast.
BT1 1LT. 0232 226434.

OXFORDSHIRE

SCIENCE STUDIO
7 Little Clarendon, Oxford
OX1 2HP. 0865 54022.

SURREY

ANIROG COMPUTERS
8 High Street, Horley.
02934 6083.

COMPUTASOLVE
8 Central Parade, St. Marks Hill,
Surbiton. 01-390 5135.

COMPUTERAMA
17 Norkway, Banstead. Surrey.
Burgh Heath 54717.

SNOW-BEECH
1 East Grinstead Road,
Lingford, Surrey.
0342 832476.

THE COMPUTER-WAY LTD.
73B North Street,
Guildford, Surrey. 0483 62626

SUSSEX

WORTHING COMPUTERS
32 Liverpool Road, Worthing,
W. Sussex. 0903 210861.

WALES

MICRO-CARE COMPUTING LTD.
18 Baneswell Rd., Newport,
Gwent NPT 4BP. 0633 50482.

TRYFAN COMPUTERS LTD.
57 Madoc Street,
Llandudno, Gwynedd.
0492 70802.

TYNE & WEAR

THE COMPUSHOP
10 Newgate Centre,
Newcastle-upon-Tyne NE1 5RE.
0632 618673.

WARWICKSHIRE

IMPULSE MICRO SYSTEMS LTD.
6 Central Chambers, Cooks Alley,
Wood Street, Stratford-Upon-Avon.
0789 295819.

W. MIDLANDS

CALISTO COMPUTERS
119 John Bright Street,
Birmingham B1 1BE.
021-632 6458

WORC[illegible]

EVESHAM COM[illegible] CENTRE
Crown Court Yard,
Evesham. 0386 48635.

YORKSHIRE

COM-TEC
6 Eastgate, Barnsley,
South Yorkshire. 0226 46972.

EMPIRE ELECTRO CENTRES
783-789 Leeds Road, Bradford
BD3 8BY. 0274 662476.

HARROGATE COMPUTER CENTRE
18 Cheltenham Parade,
Harrogate. N. Yorkshire.
04 23 57126.

TOWERLIGHT LTD.
7 Crown Street, Hebden Bridge,
W. York HX7 8EH.
0422 843520.

COMPUTERS FOR ALL
DEALERS

your ROBOT

BRITAINS FIRST ROBOTICS MAGAZINE WINTER 1984

Economatics BBC Buggy on test

ROBOT ARM MICROGRASP

Full construction details inside

A history of robotics Robots of the future

Robotics news and product guide

EDITORIAL

Welcome to the first issue of **Your Robot.** The magazine is a new publication that will appear quarterly and be presented free to readers of *Electronics & Computing Monthly.* **Your Robot** is prepared by the same staff that bring you *E&CM,* **Your Robot** being a natural extension to the coverage at present provided by *E&CM.* In fact we could have called the new magazine Electronics, Computing & Mechanics, as at present that's just what robotics is all about. We opted though for the shorter, more manageable title.

Back in time

The current state of the low cost robotics market has a lot in common with the early days of the home computer market itself. As one who was closely involved with the launch of one of the first consumer computer magazines, I can well remember the state of the industry back around 1978.

At that time the first home computers were starting to make an appearance, with the launches of the original PET, the TRS 80 and the NASCOM all taking place within a short space of time. At that time though, while the products were undoubtedly there, the unanswered question was what were they going to be used for. To most people's minds, personal computers were a prime example of a product in desperate need of an application.

The applications though followed very quickly and today there are certainly no shortage of applications for the home micro.

Robotics today

Current robot systems, at least when it comes to their use in the home, are similarly products in search of applications. There can be no doubt though that the applications will follow and that the pace of progress, or at least change in this field will be rapid.

Articles in this issue of **Your Robot** show that present industrial applications abound and also point the way to likely progress in robots for the home.

Your Robot will set out to monitor all aspects of robotics, from arms to vision detectors, from speech recognition systems to the development of software for robot systems.

If you want to keep up with this new, exciting and rapidly changing field make sure you read **Your Robot.**

Next issue – presented free with the May *Electronics & Computing Monthly.*

Editor: Gary Evans
Assistant Editor: William Owen
Contributing Editors: David Buckley,
Gary Herman, Peter Matthews,
Richard Moyle

Contents

Advertisers Index

ON THE MARKET

A run-down of the microrobotics products available today.

COLNE ROBOTICS

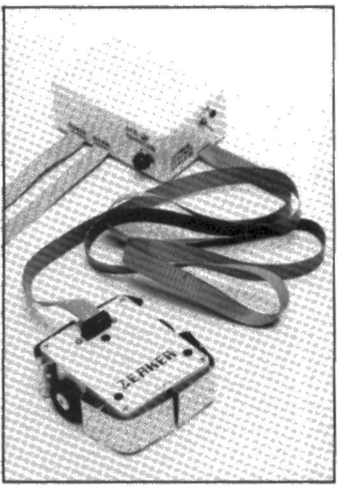
One of the first companies into the Micro-robotics field, Colne manufacture a low-cost robot arm, a turtle, fully automated robot lathes, and a low-cost robot vision system. Colne Robotics Co. Ltd., Beaufort Road, off Richmond Road, East Twickenham, Middlesex TW1 2PQ. Tel: 01-892 8197.

Armdroid 1

A robot arm with 5 axes of rotation and gripper, available assembled or in kit form. It is a continuous path machine with six stepper motors and open loop control (resolution is 2mm). The arm can be controlled by any microcomputer with an 8-bit parallel port; the software includes learning programs for a wide range of micros. Load capacity is 300g; reach is 340mm. Price is £400. (A more sophisticated arm, the Armdroid II, is available for £1500).

Colvis Vision System

A 32 x 32 pixel solid state camera which is light enough to be mounted on a micro-robotic arm, with dedicated microcomputer. Eleven feature extraction algorithms are employed and the system can recognise up to 8 objects at a time. The system may be used alone (it is programmable in its own language) or as an intelligent peripheral to any micro with an 8-bit bidirectional port.



Zeaker Turtle

As its name implies, a small mobile robot. The Zeaker has two DC motor drives, and can be controlled directly from the BBC micro or the ZX81 and Spectrum with suitable interfaces. The micro-turtle can be assembled from a kit or bought ready assembled. A computer controlled retractable pen traces the robot's path and, with the appropriate software, draws Logo graphics. The machine is supplied with 2m umbilical cable supplying control signals and power. Software is available for the BBC micro and Sinclair ZX81 and Spectrum: this enables the machine to use information from its six touch sensors.

COMMOTION

Manufacture a range of control interfaces, software, servos and mains control equipment. Commotion, 241 Green Street, Enfield, Middlesex EN3 7SJ. Tel: 01-804 1378.

Beasty

Is a control interface for the BBC micro capable of operating up to four servos simultaneously, from the keyboard. The Beasty is supplied with connection cables, demonstration program and instructions. For those who wish to program the servos, the machine code driver occupies less than 256 bytes of relocatable code (and so can be stored at any location in RAM). Unit price is £49.95.

BBC control ROM

Commotion supply the SJ Research control ROM, which plugs directly onto the BBC sideways ROM board. The 8K ROM provides simple robot control routines for inexperienced users, commands available to any language to support file handling (BASIC, BCPL, Pascal, LISP etc), software for second processors, extra I/O channels. Unit price is £44.85.

Servo Drives

A range of precision geared motors with feedback mechanism to give positional information to the computer. The standard servo (FP-S128) develops 3.5Kg/cm torque with a 100 degree range. Heavy duty and lightweight devices are also available over price range from £13 to £70.

Turn to page 8

ROBOTS – Past, Present & Future

Gary Herman reveals the subtleties of the field of robotics, and dismisses some of the fantasies.

Robotics is a subject riddled with irrelevancies, misconceptions and mundane achievements masquerading as micracles. Alternatively, it is an enterprise brilliantly illuminated by possibilities, dreams and miracles become commonplace. In short, robotics is an inconsistent thing. Not surprisingly, considering it is the bastard child of science fiction and a science called cybernetics, robotics is the repository of both wild-eyed fantasies and sober applications of what the founder of cybernetics, Norbert Wiener, called control and communication in animals and machines.

Wiener himself dates the coining of the word 'cybernetics' (from the Greek for 'steersman') from 1947. The subject itself is somewhat older with roots in work done in the thirties and early forties on signal processing, neurophysiology, statistics and computing. But the word 'robot' predates cybernetics by some two decades. It was first used to mean non-human workers by the Czech writer Karel Capek in his 1920 play 'R.U.R.' (Rossum's Universal Robots). In Czech, the word means a slave labourer. Capek's robots, however, were not electronic — they were of organic material like Mary Shelley's Frankenstein-monster. Indeed, the dream of artificial humanoid creatures flourished long before the electronics age: the myth of the clay Golem, animated by incantation, dates from the middle ages; clockwork gave rise to a rash of automata displayed throughout the courts of Europe in the eighteenth century.

But fanciful monsters and mere simulacra — no matter how superficially human-like — are not robots. Indeed, a physical resemblance to human beings is not a required characteristic of robots. Rather, the important similarities are functional: if a robot arm mimics the appearance of a human arm, that is only because human arms are reasonably efficient at what they do. Even so, a question remains: is function a sufficient criterion for determining what is and what is not a robot?

"...even the most sophisticated industrial robots rely on crude sensors..."

For some, each new machine replacing a human being (or other animal) in a specific context has been a landmark in robotics. When policemen on point duty were first replaced by traffic lights, the popular press hailed the new devices as robots. In some ways they were: traffic lights are automatic, they are functionally equivalent to some very simple human behaviour and, in theory, they are programmable (which is to say, the sequence of actions they perform can be altered). But if these three criteria define a robot, then we are already surrounded by the things — the washing machines at home, the automatic trains on the London underground, the photocopiers in the office, the jukeboxes in the pub.

In fact, there is an almost seamless divide between automated devices like traffic lights and true robots, but there **is** a divide.

The key concept is adaptivity, and adaptivity is closely related to programmability. It's a short step from a machine which simply repeats the same task over and over to a machine which can be pre-programmed to perform a number of different tasks. Pre-programmable machines are adaptable, but they are not yet adaptive. Adaptivity is achieved when the machine itself can adjust its behaviour in real time. To do this, a machine needs some input from the environment in which it is operating.

"... the subject has its roots in work done in the thirties..."

The simplest adaptive machines have been around for a long time and were, in fact, one of the main inspirations behind the development of the science of cybernetics. Speed-governors in engines (investigated in 1865 by James Clerk Maxwell) belong to the class of such machines; so do thermostats. These devices are called servomechanisms and employ simple feedback loops to adjust their behaviour towards the achievement of some particular goal.

A thermostat, for example, may be attached to a heating element in an electric boiler **(Figure 1).** It measures the temperature of the water and **feeds** this information **back** to the heating element. The boiler is set to a given target temperature and so long as the water is below that temperature, feedback from the thermostat keeps the heating element switched on. As soon as the water temperature exceeds the target level, the thermostat turns the element off. Clearly, such a servomechanism adapts its behaviour to its changing environment.

The thermostat is a painfully simple device. It has only two possible response states for an infinite range of possible environmental states. A true robot should be considerably more adaptive than that, although it could be organised (in the extreme cases) either as a monolithic multi-state device or as a hierarchical structure of two-state devices like thermostats **(Figure 2).**

A servomechanism, it should be noted, is the classic case of closed-loop control — so called because control information is effectively transmitted round a closed network. Open loop control is also possible — pre-programming being an example of this. Most actual robots use a combination of open- and closed-loop control so that, for example, an arm is coarsely positioned using digitally controlled stepper motors (these are motors which rotate in discrete steps under the control of a pulsed signal) and then finely positioned using servomotors linked to sensors.

Closed-loops can operate at any level and a system's adaptivity can be gauged by the levels at which such loops operate **(Figure 3).** A learning system must have a high-level closed-loop so that new data and events do not just alter the system's immediate responses but are integrated into supervisory mechanisms or software. Learning, adaptivity and the existence of closed-loops are, in effect, the same thing — we call it intelligence. Clearly, any intelligent system (human or machine) must be able to communicate with the outside world, for which it needs sensors and suitable software to interpret and utilise sensory data. Ideally, it should have some form of language which, once again, demands appropriate software. So a true robot needs to be more than a functional machine, and it is this 'more' that causes all the problems.

Even the most sophisticated commercially-available industrial robots — the sort that cost upwards of $100,000 and make Fiat cars — rely on very crude sensors. Most assembly operations require little more than force and pressure sensors fed

"... most industrial robots are little more advanced than a traffic light..."

through analog-to-digital converters to the control computer. Likewise, production-line tasks can be accomplished using binary visual sensors able — usually under special illumination — only to distinguish light from dark. More sophisticated sensory devices do exist — including prototype artificial skin which may approach the 20,000 nerve-endings-to-a-fingertip resolution of human touch — but for real-time applications these are just too slow. They are also too expensive and lack suitable software.

Sophisticated visual sensing is also possible — but at a price in terms of memory and speed. Gray-scale systems, light-striping and triangulation can process, typically, 256 x 256 pixel images with 256 levels of grey and extract three-dimensional information, but to run in real-time such systems need to operate at around 40 million instructions per second (MIPS) using banks of high-speed processors and very elaborate software **(Figure 4).** Charge-coupled devices are becoming increasingly important in these applications, but even the cheapest come at around $2000 for a basic sensor (such as Honeywell's Visitronic HDS-23 device) without software. Ordinary video cameras using image-processing software (edge-detection, motion-detection and pattern-recognition) are too slow for most industrial applications.

It's hardly surprising that an estimated 60% of the 60,000 or so industrial robots now said to populate the world are used to perform routine (and unpleasant) tasks on already highly-automated production lines in automobile factories. Welding and spray-painting are jobs requiring only moderate accuracy and sensitivity and the automated producted line takes most of the sweat out of positioning components to be worked on. Despite the headlines, it was not sinister when a worker was killed by a robot at the Kawasaki plant in Japan a couple of years ago. There was no more hostility involved in this tragedy than had the worker been killed by a combine harvester. Indeed, most industrial robots are technologically little more advanced than the traffic light.

Really intelligent machines are possible. They could free us for more rewarding work and be boons to the old, inform and handicapped. However, the major robot manufacturers (like the major computer manufacturers before them) will almost certainly abdicate responsibility for a socially useful robotics, leaving such developments to the same sort of amateurs and hobbyists who have blazed trails in microcomputing. With the widespread availability of cheap microprocessors and microcomputers and the growing availability of low-priced peripherals for artificial intelligence and robotics applications, it seems certain that hobbyists will once again come to the fore in a high technology enterprise.

■

"... certain that hobbyists will come to the fore in a high technology enterprise..."

NEXT MONTH:
A closer look at some of today's robots.

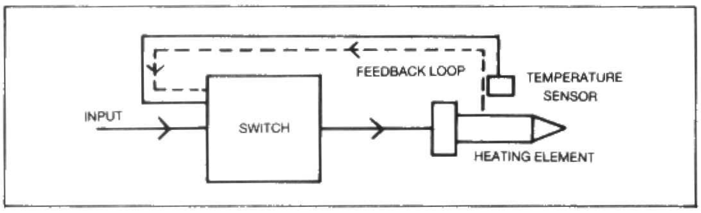


Figure 1. A thermostat is an example of a simple servo mechanism.

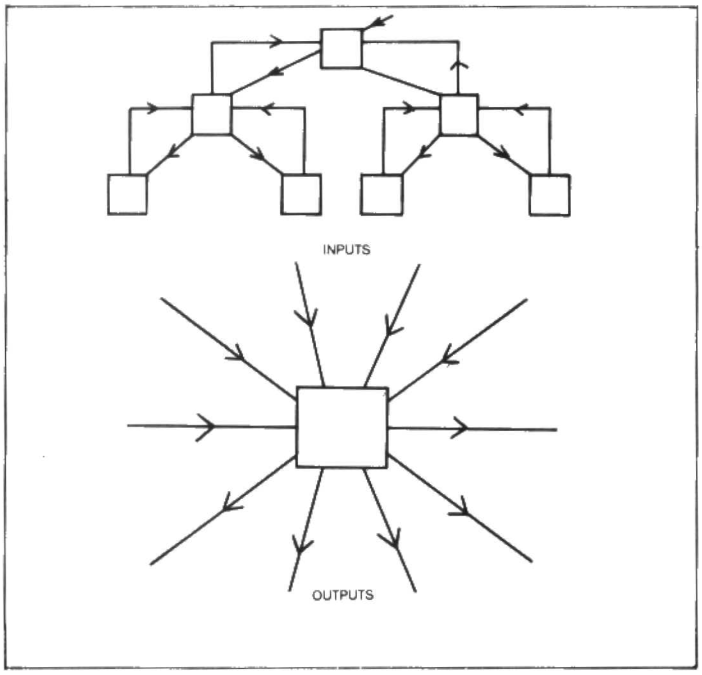


Figure 2. A true robot can be organised either as a monolithic multi-state device or as a hierarchical structure of two state devices.

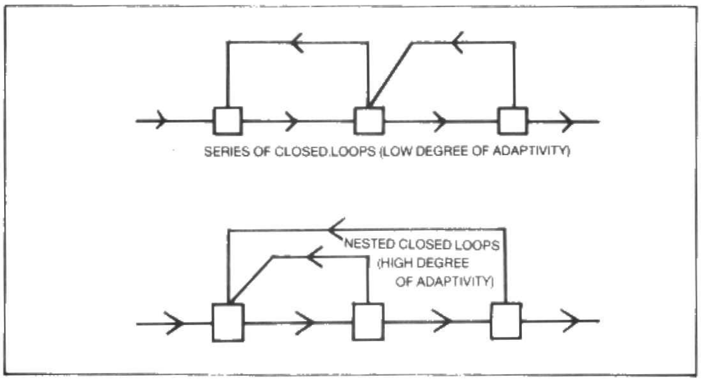

SERIES OF CLOSED LOOPS (LOW DEGREE OF ADAPTIVITY)
NESTED CLOSED LOOPS (HIGH DEGREE OF ADAPTIVITY)

Figure 3. Closed loop systems can operate at many levels – a systems sensitivity depends at the levels at which it operates.

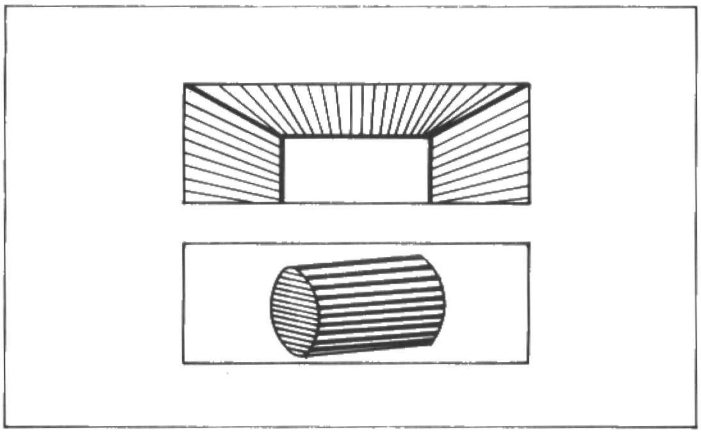


Figure 4. Sophisticated edge detection systems running in real time must operate at around 40 MIPS.

Micrograsp

Powertran's Micrograsp kit* allows a sophisticated robot arm to be assembled without recourse to complicated metal bashing.

In the field of mechanical robotics devices, the word hardware takes on its more traditional meaning, namely a world refering to a lot of nuts, bolts, and metal. Someone setting out to design and build their own robot arm would do well to have a BSc in Mechanical Engineering behind them.

The availability of kits of parts for such arms is thus very welcome and the Powertran Micrograsp is just such a kit. The basic version of the arm has an articulated arm jointed at shoulder, elbow and wrist positions. The entire arm rotates about the base and there is a motor driven gripper. Each of the arm movements is servo controlled, this servo action being independent of the computer, greatly simplifying the software to drive the robot and all programming is carried out with a small number of Basic commands.

Mechanical Operation

For each of the five axes there is a motor with integral gearbox. For the wrist and gripper motors small in-line gearboxes are used. The other axes use more powerful gearboxes in heavy duty zinc alloy castings. The shoulder and elbow joints are driven directly from the motors gearboxes with both motors mounted on the lower arm **(Figure 1).** On the upper arm and the shoulder support bracket are steel bushes clamping the gearbox shaft so that when the motors are driven there is relative movement between the lower arm and the upper arm and the support bracket.

Also on the lower arm are the position sensing potentiometers. On the bushes of these are plastic bushes on which the arm rotates. The shaft of each potentiometer is held in the steel bush fitted to the upper arm or support bracket.

For rotating the arm (axis 0) it is not possible to have the gear box shaft, the potentiometer and the shoulder support bracket all in line so the power is taken from the gearbox **(Figure 2)** by a pair of spur gears which being of 2:1 ratio result in a doubling of torque. For this axis the gearbox shaft is taken out from the motor side of the gearbox.

For raising and lowering the wrist (axis 3) the gearbox shaft rotates a bar to which the potentiometer shaft and gripper mounting plate are fitted **(Figure 3).** Through this bar the drive shaft for the gripper lead screw passes. When the screw turns clockwise the drive nut is moved in pulling the jaws together **(Figure 4).** After turning anticlockwise the drive nut moves out and the springs pull the jaws apart.

The arms for the robot each have counterbalance weights so that no voltage needs to be applied to the motors to hold the arm in the desired position. This improves the accuracy of the servoing. Without balancing an error signal would always be required for the arm to be motionless and a considerable torque would also have to be provided by the gearboxes causing an undue strain upon them.

Assembly

Construction of the robot is straightforward but the order of assembly is fairly important particularly when putting the arms together. Also if it is not to tear apart its wiring then the recommended wiring scheme should be followed.

To assist with this, holes are provided at strategic points for anchoring the wiring with cable ties.

A good starting point is the base plate on which are fitted the power supply and the rotation position sensing potentiometer. Next take off the cover of one of the gearboxes turn round the exit side of the drive shaft, fit on it the smaller gear together with its mounting bush and nut, fit the motor loosely on the top plate, screw the side panels onto the base and fit the top plate. Special screws which roll threads in the steel in which they engage are used on all the panels. The shoulder rotation shaft and the larger gear can now be fitted and the motor tightened in position keeping the gears firmly enmeshed.

The arms are constructed in a sandwich arrangement with the lower arm sides fitting round the upper arm and the shoul-

***Based on material originally published in Practical Electronics.**

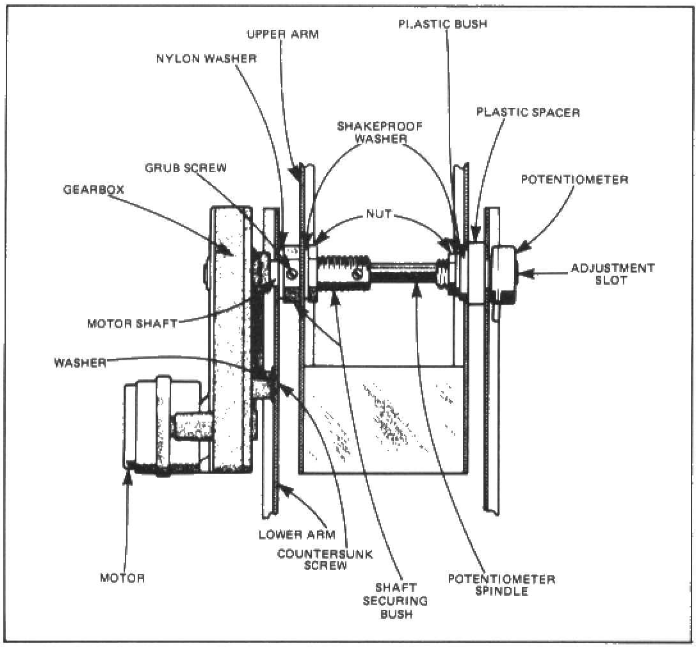


Figure 1. Detail of the shoulder and lower arm assemblies.

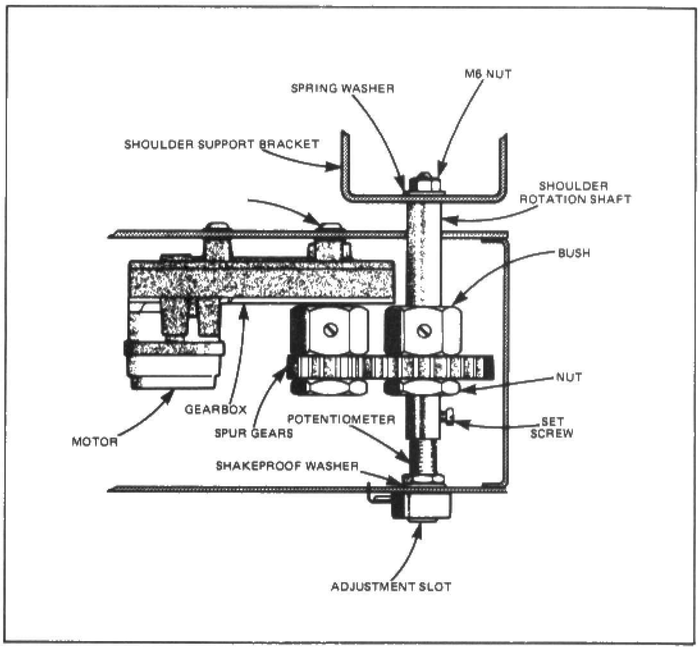


Figure 2. Arrangement of the rotating arm drive motor and gear chain.

der support bracket. First assemble the gripper components ie the jaws, gripper mounting plate, motor etc. as shown in **Figure 4.** On one of the upper arm side pieces fit the wrist motor and a shaft securing bush and on the other side piece fit the wrist position sensing potentiometer. The two sides can now be brought together sandwiching the counterbalance weight and the gripper assembly. Next fit the shoulder support bracket, screw on it the other shaft securing bush and fit to the lower arm side pieces the motors, counterbalance weights and potentiometers.

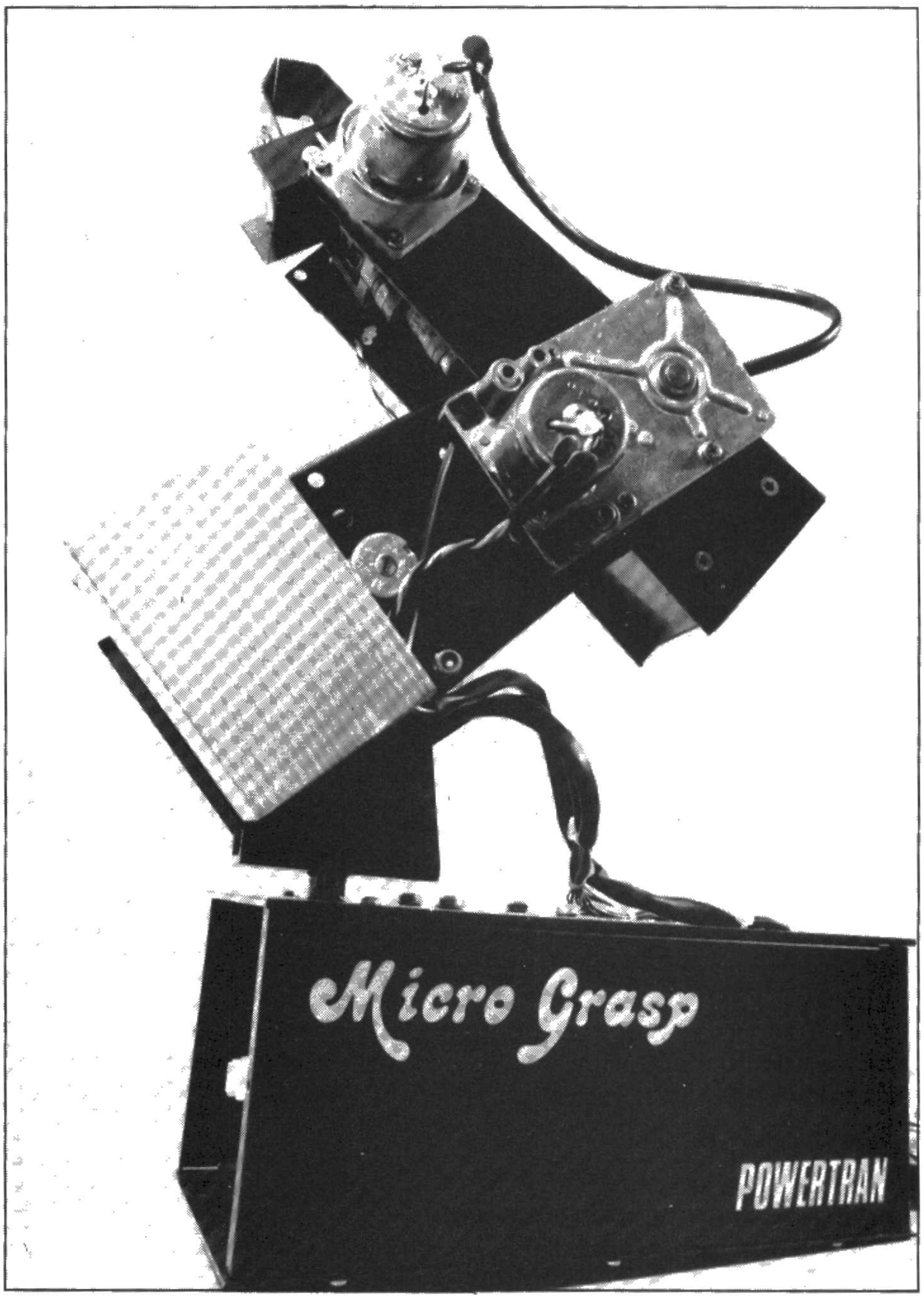
The side pieces can now be brought together round the upper arm assembly and the shoulder support bracket, holding them together with a stud through them at what will be the rear end of the machine. The wrist motor is on the right hand side. Secure the gear box drive shafts but not the potentiometer shafts and move axis 0, 2, 3 to the centre of their travel, axis 1 to just below its limit — ie gripper, upper arm and lower arm all in line about 20° below the horizontal with the arm pointing forwards.

Set each potentiometer except the lower arm potentiometer which is set to fully anticlockwise to its centre position ie equal resistance between the centre tag and each of the outer ones by use of a screwdriver in the adjustment slot and secure the shafts.

The robot can now be wired up to terminal blocks fitted to the rear end panel. The wires in the 6mm sleeve pass through holes in the bottom of the shoulder support bracket before passing through grommets in the top plate. Sufficient slack must be allowed for 180° of movement of the arm.

In Control

With mechanical construction of the arm complete, we turn to the subject of controlling the Micrograsp. For each of its axes of movement, the Micrograsp requires a control voltage to be applied to the appropriate



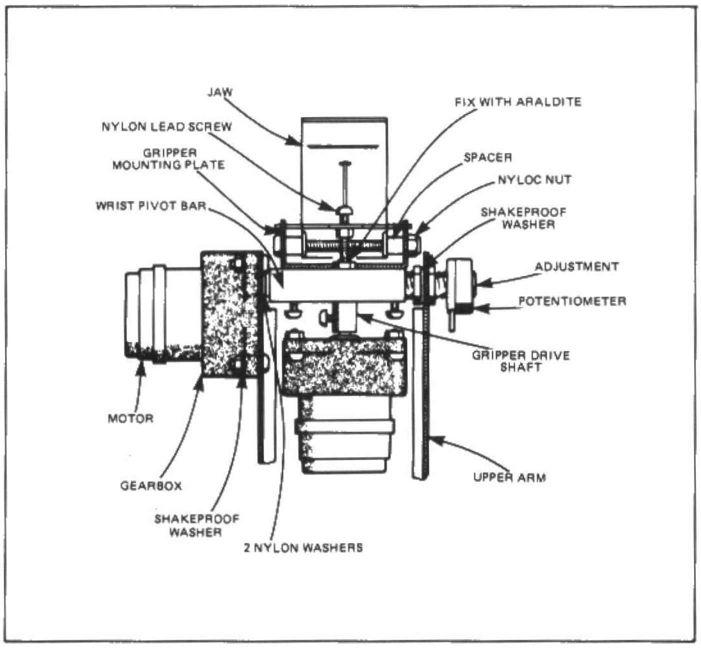
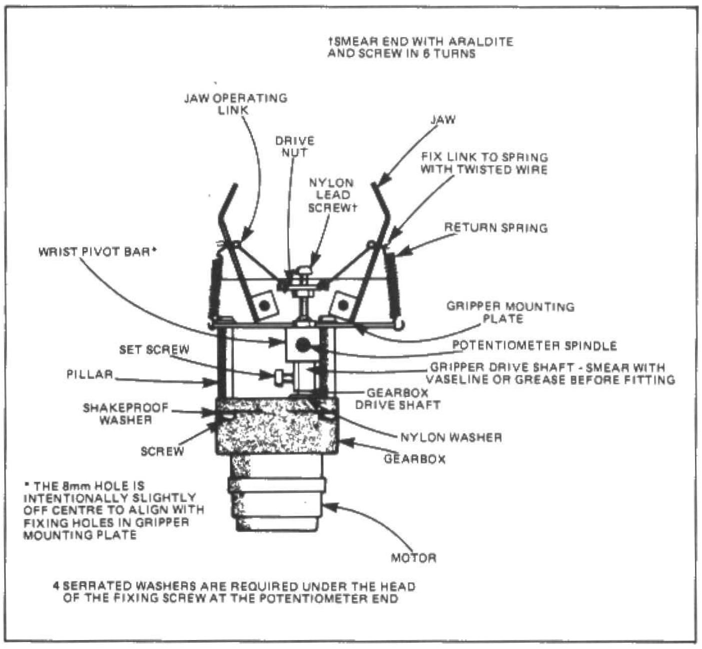


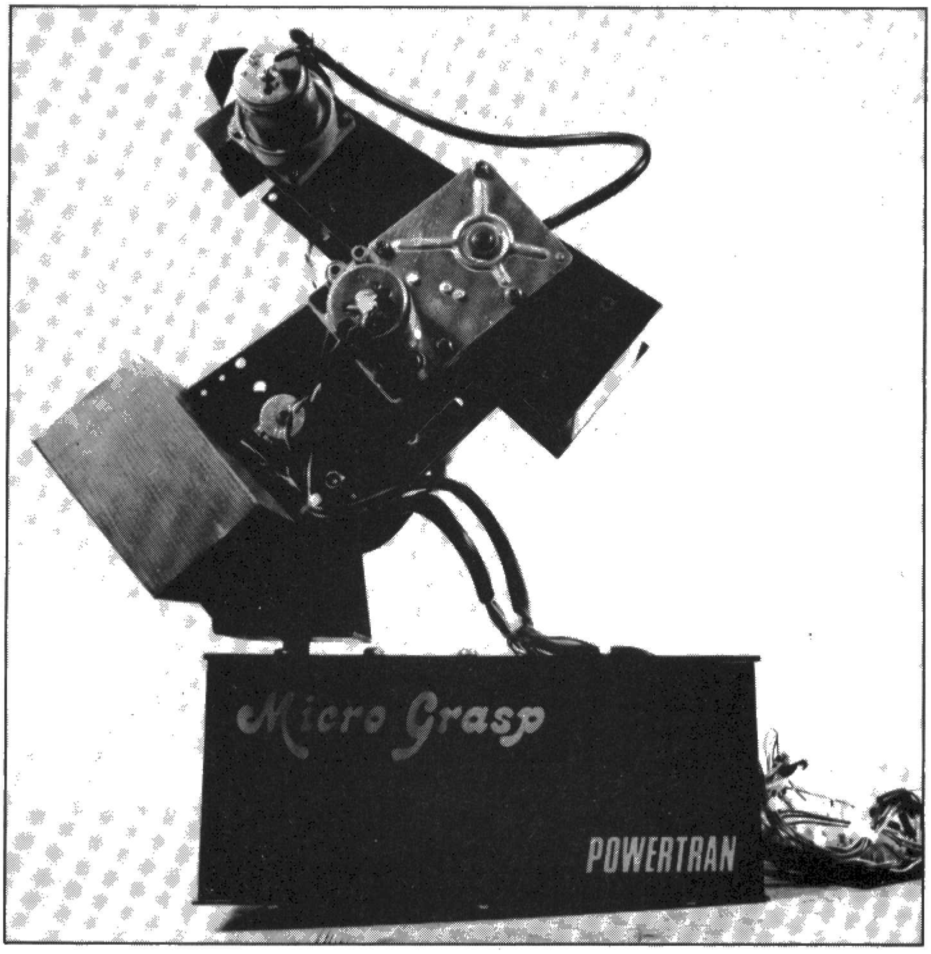
Figure 3. The pair of motors associated with the wrist assembly.


†SMEAR END WITH ARALDITE AND SCREW IN 6 TURNS

* THE 8mm HOLE IS INTENTIONALLY SLIGHTLY OFF CENTRE TO ALIGN WITH FIXING HOLES IN GRIPPER MOUNTING PLATE

4 SERRATED WASHERS ARE REQUIRED UNDER THE HEAD OF THE FIXING SCREW AT THE POTENTIOMETER END

Figure 4. The gripper assembly with its lead screw.



motor, this voltage being dependent upon the signal appearing at the wiper of the motor's associated potentiometer.

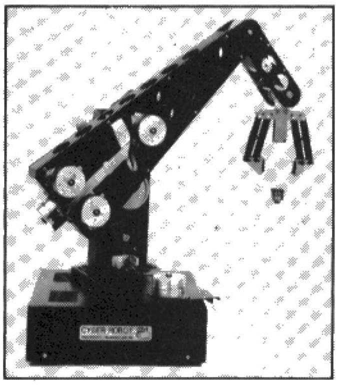
One approach to control of the arm would be to adopt the hardware intensive approach and to dedicate an analogue bridge type amplifier configuration to each motor and to drive the amplifiers with a difference signal from the appropriate potentiometer and a voltage derived from a binary value poked into an eight bit latch combined with a D/A converter. Powertran supply such a board which can be used with a ZX81 and indeed any computer providing basic I/O facilities. We shall return to this board in a future issue of **Your Robot.**

An alternative approach, and one that will appeal to owners of the BBC model B micro, is to use a more software orientated scheme. Here two motors would be controlled by Pulse Width Modulation Techniques. PWM has, for a number of PWM system, the motor is fed with a digital drive signal, the motor being off if fed with logic 0 and running at full speed when logic 1 is present. By rapidly changing the voltage applied to the motor, and altering the ratio of the lengths of the 1 and 0 signals it is possible to run the motor at any desired speed. PWM also has the advantage of providing a high torque at low speeds.

Any computer is an ideal PWM controller, with software delay loops determining the motor's speed, the motor being driven from a power transistor on one of the computer's data lines. ■

ON THE MARKET

cont. from page 2



CYBER ROBOTICS

Manufacturers of the Cyber 310 and Cyber 3 robot arms, speech synthesis systems, and the recently announced Cyber 410, a robot designed on a similar basis to the human arm. Cyber Robotics Ltd., 61 Ditton Walk, Cambridge CB5 8QD.

Cyber 310

A redesigned version of the Cyber 3, the 310 is an educational robot arm with five degrees of movement plus gripper action. This model has the ability to rotate the shoulder 300 degrees in the vertical plane to operate on the opposite side, in effect with the arm upside-down. FORTH is used for the operating system. Simple BASIC routines are also included.

ECONOMATICS (EDUCATION)

Economatics manufactures the most famous robotic device around, the BBC Buggy. The Buggy is Economatics first venture in the robotics field. Economatics (Education) Ltd., 4 Orgreave Crescent, Dore House Industrial Estate, Handsworth, Sheffield S13 9NQ. Tel: 0742 690801.

BBC Buggy

The Buggy is a turtle type vehicle constructed from Fischer Technik. The Buggy is intended for educational use alongside the BBC computer course, and the programs provided with the kit are designed to take the user step by step through the processes of computer control and switching, gathering information, solving problems etc. The kit can be expanded and enlarged using extra motors and building blocks.

JESSOP MICROELECTRONICS

Manufacturers of the Edinburgh turtle. Jessop Microelectronics Ltd., Unit 5, 7 Long Street, London E2 8HN. Tel: 01-739 3232.

Edinburgh Turtle

A two wheel, single motor device mounted under a transparent perspex dome. The feedback system is a photoelectric cell and light beam which is broken by each revolution of the wheel. A centrally mounted pen can be raised or lowered by computer control (by Logo). The pack comprises the Turtle, interface set, (including cables), software on disc or cassette, pens and documentation. Software is available for a wide range of home computers. The price is £145.

POWERTRAN

Manufacturer of the Micrograsp, one of the cheapest robot arms on the market, and of the Genesis range of hydraulic robot arms. Powertran Ltd., Portway Industrial Estate, Andover, Hants SP10 3WN.

Micrograsp

An articulated arm jointed at the shoulder, elbow and wrist, and driven by any simple home computer such as the ZX81, Spectrum or BBC. The gripper is motor driven, and each of the arm movements is servo controlled, ie position sensors feedback information to the computer. There are five axes. Price is £145.

Genesis

A representation of an industrial robot arm with hydraulic operation. The basic system can be controlled by the teaching pendant giving complete manual control over all 5 axes of movement and the grip. Interfaces are available for the BBC, RS380Z, Apple II, Pet 3000, AIM65, TRS80, ABC80 and MAT385 computers.

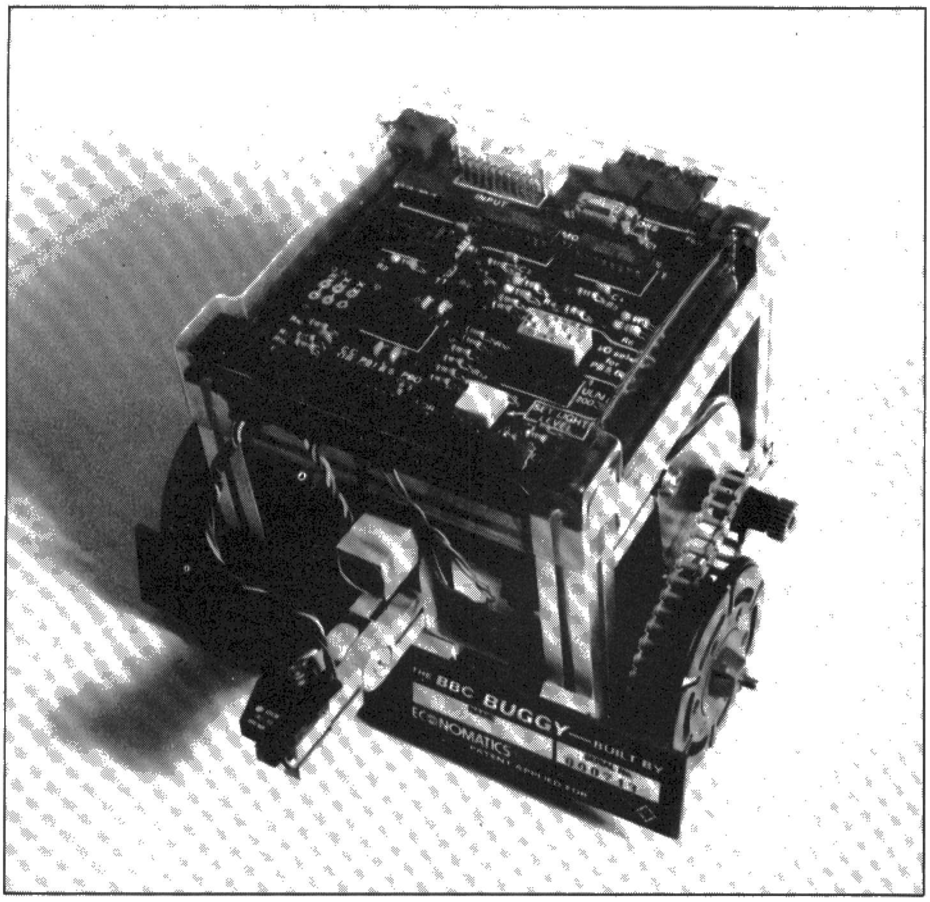
SYKE INSTRUMENTATION

UK distributor of Microbot (USA) Minimover and Teachmover robot arms. Syke Instrumentation Co. Ltd., Syke House 117/119 Station Road, Liss, Hants GU33 7AJ.

Minimover and Teachmover

Two robot arms designed for education in industry and colleges, they are in fact termed training robots. Both offer more precision, better machining and durability than those arms in the £200 range (ie the majority). The price is around £1,000.

Syke have also announced the introduction of the SYKErobot, designed to replicate as near as possible the movements of the human arm, for use in such environments as food processing and laboratory use. The new arm is capable of handling weights of up to 2Kg.



Featured as part of the BBC's computer literacy programme, Economatics' buggy is an ideal introduction to the world of Artificial Intelligence. Peter Luke takes the Buggy for a test drive.

ECONOMATICS' BBC BUGGY

The literature accompanying the Economatics' Buggy opens with the phrase "Welcome to the World of Artificial Intelligence". Readers of Mike James' series on AI (the Computer Brain, January 1983 – August 1983) will be familiar with the many theoretical aspects of the field — the Buggy provides an introduction to the more practical side of AI.

The Buggy is based on a chassis constructed from standard Fischertechnick parts together with a couple of stepper motors, circuit boards and various cables and connectors. It is available as either a kit or as a ready-built assembly. The Buggy supplied for review was the assembled version and thus we are unable to comment and the process of putting the kit together. Having said that though, construction should not pose any problems, the only tools required are a screwdriver and M4 spanner, and going through the assembly process would appear to give a good understanding of the way in which the various sections of the Buggy combine to form the finished unit. The assembly and operating manual contains full details of the construction procedure and makes interesting reading even if the fully assembled version is purchased.

Wiring up

The Buggy's main circuit board is connected to an interface board via a generous length of 20 way ribbon cable. The interface board in turn is connected to both the user and analogue ports of the BBC micro. A third connection is also made to the DC power outlet of the computer. The manual warns that the BBC micro's analogue port has no protection against static discharge nor for out of range inputs. The interface board makes up for these shortcomings however, and eliminates the risk of any damage to the computer when using the Buggy.

Features

The Buggy's frame, in addition to the two stepper motors providing individual drive to the wheels, houses various tactile and optical sensors. Tactile sensing is realised by a pair of large bumpers at the front of the robot which close one of two microswitches if the Buggy encounters an object. Optical sensing takes place both in the visible and infra red with an LDR sensor and a pair of semi-conductor devices that form an infra red bar code sensor.

TABLE 1
TITLE*
TEST
SWITCH
MSWITCH
RPLAN
RECORDER
SNAIL
EXWALL
EXOBJ
BARPLAN
ALLEY
MVB
SUNSEEK
LINE

In addition, there is provision for a pen to be mounted on the Buggy's chassis and for the pen to be raised and lowered by a solenoid.

Software

The Buggy is supplied with a suite of software that begins with a program designed to exercise all of the Robot's facilities. These include various 'fine tuning' procedures to ensure that the Buggy travels in a straight line when told to do so and also that it's physical rotation matches that demanded by the software. Also included are tests to verify correct operation of the two bumpers, an adjustment of the sensitivity of the light detection circuitry (accomplished by way of a pre-set potentiometer) and calibration of the infra red bar code reader.

Using the software

In addition to the test routine the rest of the Buggy's software consists of programs that initially provide the user with the means to control the elementary aspects of the robot's performance and lead on to some quite sophisticated routines exploring aspects of AI.

Table 1 lists the programs available, the first of these being Switch. This allows direct control of the Buggy's motion by means of the BBC's cursor control keys. As with all of the programs on the tape, Switch can be listed and a careful study of this will demonstrate the various techniques used for control of the Buggy's locomotion.

Memory Switch builds on the concepts introduced by Switch but allows a route to be remembered as a set of segmental instructions. The segments are formed from a length (in centimetres) or a rotation (in degrees), the values being dependent on the time for which the movement controls are held down.

Routeplanner introduces an alternative means of determining the Buggy's route allowing the path to be planned on screen before execution.

Recorder introduces the concept of a Buggy Park, rectangular area that forms the Buggy's world. Snail continues with the Buggy Park theme but introduces a more sophisticated control of the robot's movement with an English Language instruction set. This consists of the following commands — forwards, backward, left, right, wait and stop. As before movements are entered in centimetres and rotation in degrees.

Explore the Wall is the first program to make use of the tactile sensors and allows the machine to map out its environment without any intervention. Explore for Object continues in a similar fashion and allows Buggy to refine the topography of its pack. Again, the program can be listed and the operation of the search algorithm investigated.

The Bar Code

Barplan introduces one of the major attractions of the Buggy, its ability to accept information in a jumbo bar code form. The simplified codes allow the Buggy to scan a set of cards that input the numbers from 1 to 9. Directional movement is now represented by the numbers 1 to 5. The Buggy scans the cards in threes, the machine interpreting each triplet as a single route instruction.

Tin Pan Alley extends the scope of the bar code reader by allowing musical notes to be read by the Buggy.

The final programs on the tape exercise the optical sensing element of the Buggy and allow the robot to search for a source of light and to follow a line.

Conclusions

The Buggy provides a valuable introduction to the field of Robotics and AI. It is both easy to connect to the BBC micro and of a rugged enough construction to withstand any of the inevitable knocks it is likely to suffer in use.

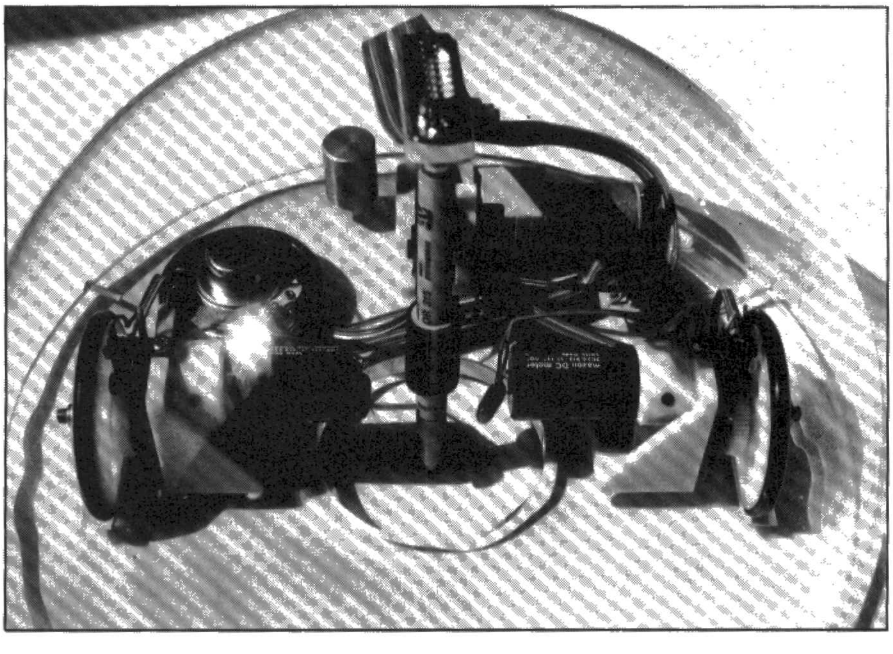
The software supplied is comprehensive and the ability to list the programs means that a user should quickly learn the control techniques involved.

Economatics
Education Division
4 Orgreave Crescent
Dore House Industrial Estate
Handsworth
Sheffield
S13 9NQ

■

THE EDINBURGH TURTLE

Or how to teach simple geometry and computing techniques.



The Jessop microelectronics Edinburgh Turtle is the production version of a machine designed by the Artificial Intelligence Department of the University of Edinburgh, specifically for use in schools as a means of teaching children of different ages the principles of computer control.

The turtle is a two wheeled vehicle driven by a single motor, covered by a perspex dome. Information about position and distance travelled is fed back to the host computer by an ingenious photo-electric cell device. With every turn of the wheel a cog breaks the beam and each revolution is thus recorded and the information returned to the computer via the umbilical. Suspended perpendicularly at the centre of the dome is a pen which traces out the movement of the turtle (the pen can be lifted from the surface over which the turtle travels by computer control).

Using simple LOGO commands any number of different shapes can be drawn by the turtle. 'F' means forward, and 'B' back: therefore F100, R90 (ie right 90°), F100,R90,F100,R90,F100 draws a square. This movement can be stored in memory, given a name (eg SQUARE) and thence repeated every time the instruction SQUARE is entered. As a further example of the languages capabilities, a house could be drawn by combining a square with a triangle, and using the instruction "Times 10 procedure House" a whole street could be drawn. The Turtle is therefore an excellent teacher of geometry as well as logic.

The turtle is available in two versions: Turtle-SERI, with an interface unit with its own power supply and control program in EPROM (a construction enabling use with any micro's serial output); and Turtle-PARA, with an interface tailormade to each type of microcomputer. The control program is loaded from the software.

The full turtle pack comprises the vehicle itself, the interface set (including cables), software on cassette or disc, 2 pens, and documentation. The turtle is priced at £145.

■

BASICARE'S LEGOID

While Lego building blocks may not seem the most obvious material from which to build a robot, Basicare have used them to good effect. Ken Alexander interviews Legoid somewhere in West London.

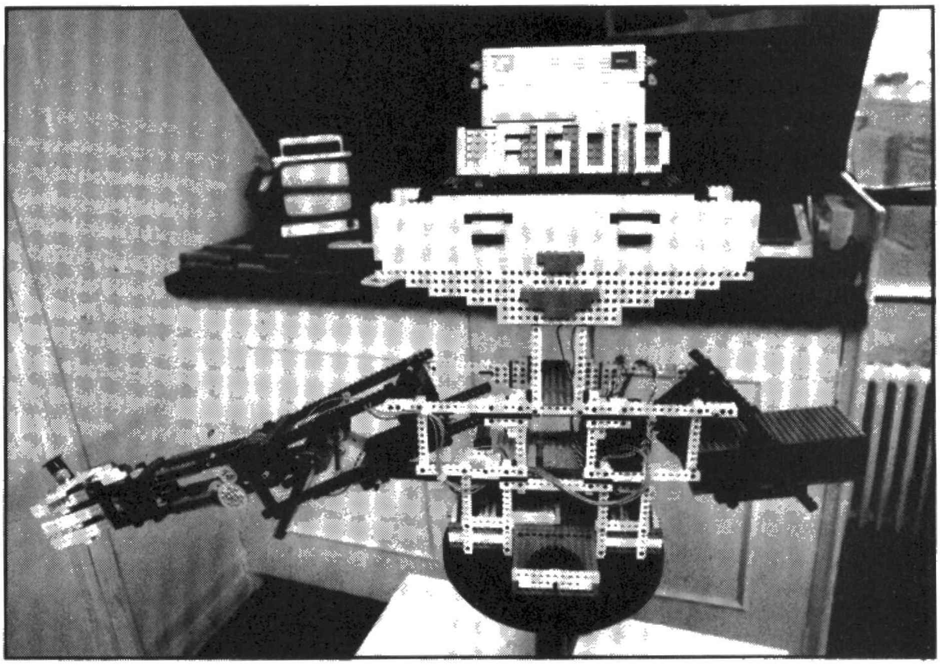


Photo 1. An overall view of the Legoid robot.

Anyone visiting Basicare's stand at last year's PCW show will have seen the company's Legoid robot in action. The robot was designed to demonstrate the use of Basicare's range of microdrive compatible Spectrum add-ons that include a relay based control unit. (The company also produce a range of ZX81 modules and a review of these together with an explanation of the concept of a persona unit appeared in the May 1983 issue of *E&CM*).

Legoid, a name that incorporates both the words ego and id — very Freudian — is constructed in the main from standard Lego components, the only 'foreign' bodies being a number of micro switches used as end of travel detectors.

Looking at the structure it would be easy to assume that liberal amounts of glue had been used to keep the robot together but in fact no adhesive was necessary, a tribute both to the lego system and to the design of the Legoid.

Outline Construction

It took Basicare just one week to build Legoid — from initial design to the finished product - although it was pointed out that the week comprised some very long nights.

The majority of the robot went together quite smoothly although it took some time to come up with the rack and pinion mechanism that eventually proved to be the solution to the problem of implementing the shoulder and elbow movement. It is hoped that our photographs give a good idea of how the various sections of the robot went together, including the excellent gripper modelled on a human hand rather than the usual crab's claw.

All the motor and gear box assemblies were standard Lego components and, as with all Lego parts, were found to be of extremely high quality.

Why Lego?

The decision to use Lego as the basis of Basicare's robot was taken primarily on the grounds that Lego is the most versatile of the similar systems available in the UK and, of almost equal importance, that Basicare managed to form a casual relationship with Lego UK.

The British arm of Lego's multinational operation supplied Basicare with a large number of technical and construction kits from which Peter Chow of Basicare selected the various items that went to make up Legoid. There is a potential difficulty here for anyone wishing to emulate Basicare's robot building work, for the various parts for the robot came from a large number of different kits and when construction was complete there were a corresponding number of pieces left over. The total cost of all the kits is very high even though the cost of those required for Legoid is far more reasonable. Unfortunately it is nigh impossible to obtain individual Lego parts and thus until Lego decide to market a robot building technical kit it will be very difficult for individuals to experiment along the lines of Basicare's work.

The chances of Lego marketing such a kit are still an unknown quantity but Basicare are hoping that the parent company will give the go ahead for such a project. If this were the case the basic arm could well sell for around £30. At this sort of price, the resulting easy to construct and versatile arm would have appeal to both the home and educational experimenter. ■

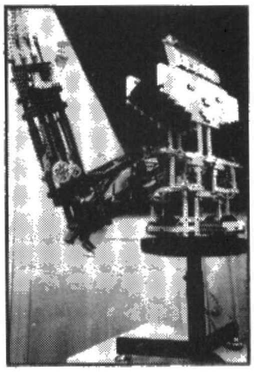


Photo 2 (right) shows an alternative view of Legoid.

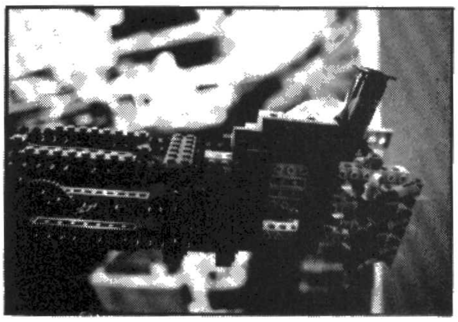


Photo 3. Close up detail of Legoid's human like grasp.

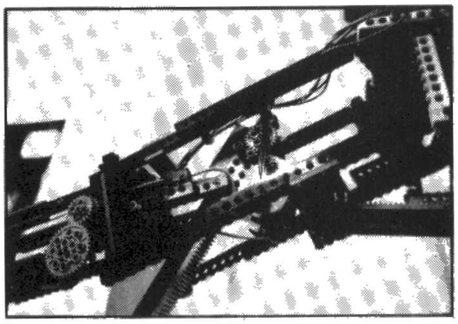


Photo 4. Showing the rack and pinion elbow joint.

MICROBOTS

Micro-robotics today has reached the stage that the microcomputer industry reached ten years ago – but the pace at which robotics is moving is much faster. Peter Matthews examines the state of the art in Microrobotics, and suggests a few possible applications for the low cost robot arm.

The market and technology of small manipulating robot arms is better developed in the UK than any other country in the world. There are more manufacturers of small robots here than the USA and Japan put together. The question is, are they tools or toys and do they have any place in industry and commerce. A part of the answer comes from the Atomic Research Establishment at Harwell. Dr. Brian Pearce who is concerned with radio active handling is examining their use for the handling of isotopes. As you might expect the exhaustive test programme that he has devised gives most of the small robots a good 'going over'. His conclusions are that most of the devices are inadequate with the exception of the American Microbot and even that passed the test by a small margin. So if small robots are to be useful tools then their performance in positionality, repeatability, speed, programmability and even robustness will have to be improved.

That, however, is not the whole answer. There are some exciting applications that the small robot has opened up by various levels of experimentation. The five thousand devices which have been sold to Universities, colleges, schools and industry at £200 or under have a wide range of applications even if the results of robot performance are not quite up to scratch.

Serious money is beginning to be spent on the next generation of small robots. What this is going to do to larger robot manufacturers is a matter of speculation but their market is going to change. It seems probable that the small robot will change the market in the same way that the mini and micro computer changed the mainframe market and technology. In fact the trends in robot market time scale is much faster than the computer's market history.

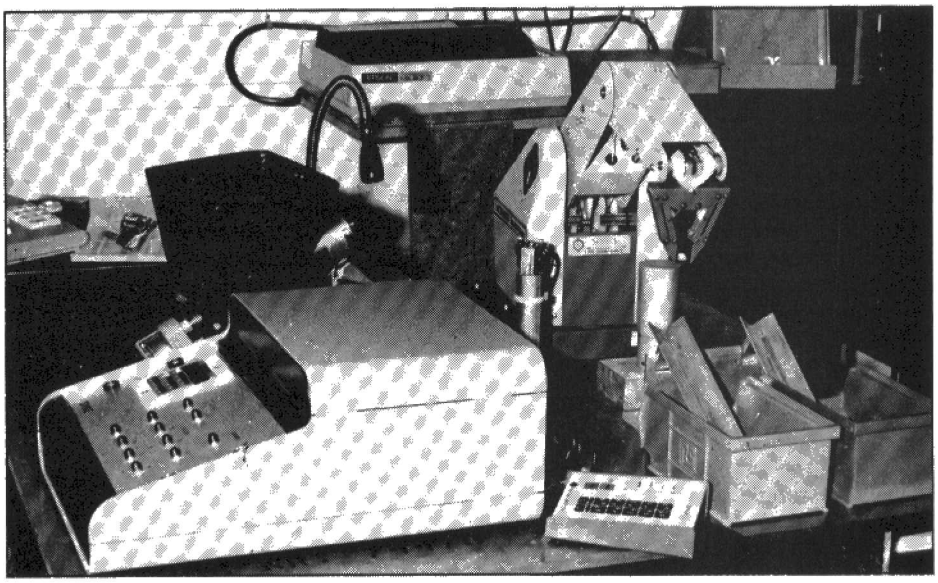
The robot market started in the 70's and a few companies such as the US based Unimation Corporation started a lonely crusade to sell and educate the industry. Slowly and at great cost individual applications were found for robots in welding and paintspraying but there were a few casualities along the way. Companies found that in addition to the cost of the robot at £30,000 to £80,000 there was an installation cost of about twice the cost of the robot. Even so it required a dedication to install the robot that some companies did not have. Some gave up and sent their robots back. This period compares to the mainframe computer market in the 1960's.

At the end of this decade the mini computer came in. This might be thought to compare to the introduction of lower cost robots like the PUMA which cost £15,000 to £20,000 in the late 1970's. At this time the cheaper, leaner, more effective robot led to the explosion of interest of the media in robotics. The huge amount of coverage baffled most Directors and production engineers in manufacturing industry. This caused the development of the microrobots as the same sort of situation led to the introduction of microcomputers in the 1970's.

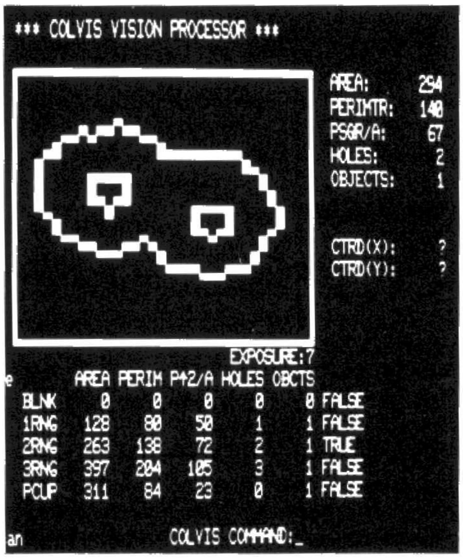
In 1980 there was only one small robot on the market and that was the American Microbot in Los Angeles. In August 1980 the Armdroid and the Genesis were both launched by magazines. Along with the Smartarm these were to be the big sellers in the next four years. The reaction of the market to the small robot was much the same as that accorded to the 4K microcomputer and not unreasonably, it was considered a toy. What the letters to the technical press from at least one Managing Director of a large robot manufacturer did not mention in its invective was that the small robot was a base for experimentation. What was more it was 'user friendly' and that made the small robot a different product to its big brother. Low cost and user friendliness allied to the fact that many small robots could be driven by standard microcomputers that schools, colleges and industry knew, and loved (or hated) took the 'black box' mystic from robots. It also began to make the customer look at the inflated price of the industrial robot, the cost of which had just begun to come down in the 1980's.

The small robot came in for some surprising uses. Doctor Gill at Brighton Polytechnic received a substantial grant from the EEC to develop a robot for spastics and the disabled.

Other dramatic uses came more slowly. The main uses were (and still are) training and education. The new A levels in computer control pioneered by Professor Chaplin in Essex University found the small robot an ideal teaching aid. Typical of the training effectiveness of small robots was illustrated in the plant of a big car manufacturer. They had installed welding robots on the production line and wanted to give some instruction and understanding of foremen and charge hands in the factory. The first attempt was with the help of the robot manufacturers. A salesman was sat down and he described the operation and usage of the robot. His computer/robot jargon left his audience cold and baffled. The management did not give up but went to a leading production engineering association and set up a basic scheme which used microrobots for instruction. The production staff were taught using a 'hands



The Syke Teachmover. A robot at work in a hazardous environment: sorting radioactive pellets.



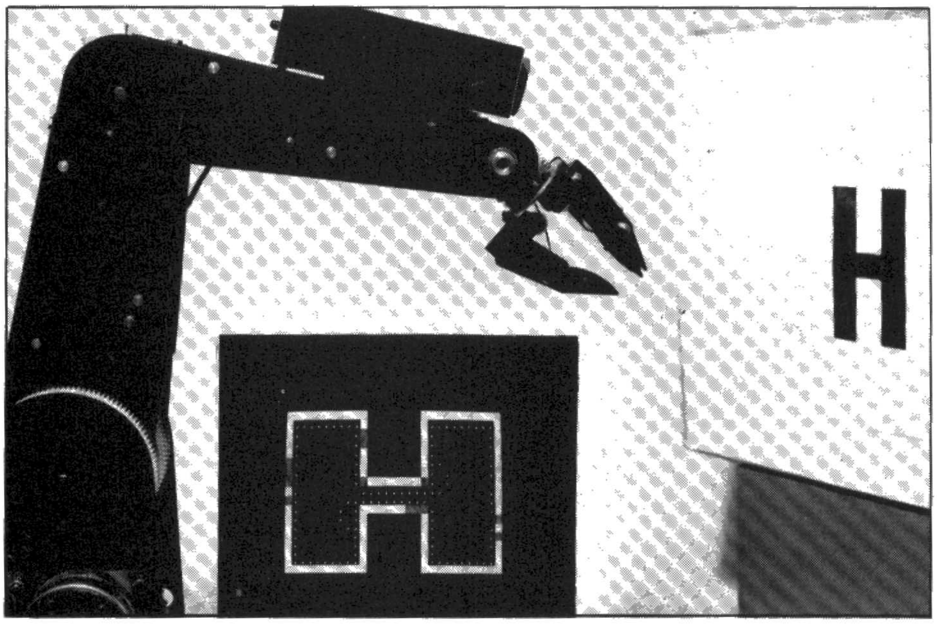
```
*** COLVIS VISION PROCESSOR ***

                                AREA:      294
                                PERIMTR:   140
                                PSQR/A:     67
                                HOLES:       2
                                OBJECTS:     1

                                CTRD(X):     ?
                                CTRD(Y):     ?

                         EXPOSURE:7
e      AREA PERIM P^2/A HOLES OBCTS
 BLNK     0     0     0     0     0 FALSE
 1RNG   128    80    50     1     1 FALSE
 2RNG   263   138    72     2     1 TRUE
 3RNG   397   204   105     3     1 FALSE
 PCUP   311    84    23     0     1 FALSE

an              COLVIS COMMAND:_
```



Two low-cost vision systems: above, Colne Robotics' Colvis system; right, the Cyclops system. The 'H' image is scanned and repeated on the VDU.

on' method. The practical experience made all the difference to their understanding. Their ability to build up and control movement sequences by operating their own robot rather than occasional use of an expensive industrial device made a considerable difference to the students and their tutors.

There is to be a major training initiative in 1984 where the OU will offer courses to industry. The courses will include a robot capable of handling several kilos. Several companies are competing for supply of a robot of the OU specification. The price is expected to be as low as £2,000 or £3,000 which will be made possible because the order for the successful bidder will be for several hundred robots. This will probably be the world's largest single order for robots.

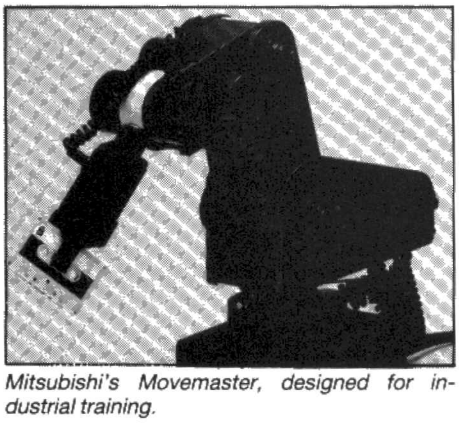
The more practical applications of the small robot have not been in the area of industry. So far the devices have not been designed as industrial tools and are not regarded as robust enough for the factory environment. There is however a move to experiment with them in laying wire looms and also the drilling of printed circuit boards. Both of these have uses in electronics but experimental use in mechanical engineering is comparatively rare and does not compare with the use of the industrial robot in mechanical engineering.

There is however one application area in which there is rich promise of the effective use of small robots. It is in the laboratory. Some say that the advent of the small robot will mean the beginning of the unmanned laboratory in the same way that the industrialised dream of the unmanned factory. Certainly the small robot seems poised to do many tasks in the laboratory when the devices have improved a bit in mechanics, electronics and software. The difficulty with the small robot however is not so much the technology of the robot but how the robot is applied to the tasks in hand.

The principle that one should know more about the technology to which the robot is being applied than about the robot itself is very sound. The Department of Industry was well aware of this when they set aside £400,000 to develop laboratory applications. The laboratory of the Government has taken the decision to be the centre of excellence in the field and has started its own robot experiments. A successful initiative to develop robots for laboratory usage would make Britain a leader in the field because although there are a great many manufacturers and applications in Japan, America and even Europe there is only one company of any importance in America that has any laboratory robot capability.

Laboratory work consists of generally small concentrations of very highly skilled people commanding high salaries who spend much of their time performing the same titrations (mix until it bangs or changes colour) hundreds of times to prove a point.

Laboratories of the future will be changed as much as factories will be by the introduction of robotics. Already robots are programmed to weigh chemicals in routine procedures particularly if those chemicals are unpleasant carcenogens. Other more complex tasks are being worked on in conjuction with more complex laboratory devices than the electronic weighing balance. The unmanned laboratory is being planned by several equipment manufacturers even now. It is however in environmental areas that there will be the greatest change and saving.



Mitsubishi's Movemaster, designed for industrial training.

Clean rooms with a sterile atmosphere have always been expensive installations. They are almost invariably used for pharmaceutical or microelectronic purposes. The manufacture or packing of certain tablets or the assembly of electronic chips are examples of common use. The robot in either its programmable form or as a device controlled from outside the room are being experimented with in British clean rooms. Some laboratories are using industrial robots. Many find they are getting problems from contamination with dust, grease, oil etc. from the mechanism. Others are having better results from small electronically driven devices. The effect on the size, use and cost effectiveness of clean rooms is going to be dramatic. The fact that operators do not have to enter the environment but can manipulate the work from outside. This means that a room becomes a cabinet or even a small tunnel of transparent material with a small robot with any controls needed on the outside. Such developments could reduce a clean room's cost from large fractions of a million to a few thousand pounds.

The applications above are thought to be realistic mainly because they are the subjects of experiments at this present time. There will be others mainly because the technology is expanding. The ability to give robots vision has already made industrial robots much smarter. The development recently of low cost vision systems of various levels of sophistication and cost is already taking place for the small robot. Tactile sensors are also going to give a new dimension to the technology. Much of this and the hardware and software to back it up are going to provide new horizons for the roboticist in colleges and industry. The field is one of considerable possibility and interest spreading into many industries and applications. The rather surprising and pleasant fact that Britain is in the lead in this technology could well offer a real opportunity in the future of robotics in many areas of light engineering. ■